# SCOOP: Scalable On-Chain-Off-Chain Storage in Blockchain-based Decentralized Identity for Cyber-Physical-Social Systems

Kai Ding, Tianxiu Xie, Keke Gai, *Senior Member, IEEE,* Jing Yu, Shuo Wang, Jianbo Gao, Liehuang Zhu, *Senior Member, IEEE,* Weizhi Meng, *Senior Member, IEEE*

**Abstract**—Blockchain-based *Decentralized Identity* (DID) technology provides a more secure and efficient paradigm for identity verification. However, along with the promotion of applications, data storage of DID has gradually become one of restrictions for the implementation due to the dramatically increasing size of the distributed ledger, and the issue becomes more complicated when considering the diversity of practical conditions and interoperability. In this paper, we propose a *Scalable On-Chain-Off-Chain Storage for Decentralized Identity* (SCOOP) scheme to minimize the total storage cost of blockchain-based DID while guaranteeing the efficiency of the implementation. The proposed scheme consists of a two-phase decision-making process and a method integrates the on-chain storage with off-chain storage in a scalable manner, so that an optimal storage task scheduling can be generated. In addition, to reduce on-chain storage overhead, we propose a multilinear tree–based commitment scheme that supports sublinear proof aggregation and updates. Our experiment evaluations have demonstrated that the proposed scheme can successfully achieve a superior performance in storage saving for DID while considering the time constraint.

**Index Terms**—Decentralized identity, on-chain-off-chain, blockchain, storage optimization, commitment

## 1 Introduction

*Decentralized Identity* (DID) has emerged as a foundational building block for user-centric authentication and authorization in distributed systems [1], [2]. In *Cyber-Physical-Social Systems* (CPSS) [3], [4], where humans, devices, and service providers continuously interact across organizational boundaries, DID-based services must simultaneously support interoperability and privacy. However, as DID deployments scale, the volume and frequency of DID-related data objects (e.g., credentials, authentication evidence, and audit records) increase rapidly, and naïvely persisting them on the blockchain leads to severe storage bloat and high operational cost. For example, in the context of smart healthcare, it aims at providing patients with medical strategies matching both treatment demands and accountability, which allows using one account to access multiple medical systems and relying on secure data sharing to obtain a higher-level comprehensive medical diagnosis [5], [6]. Moreover, CPSS nodes are inherently heterogeneous and resource-constrained; edge participants often face tight time budgets and limited battery capacity, making DID data storage a practical optimization problem rather than a purely security-centric design choice [7], [8].

A common solution to ledger bloat is to adopt hybrid on-chain/off-chain storage: storing bulky content off-chain while anchoring integrity proofs on-chain [8], [9]. Recent studies have explored blockchain and IPFS-based off-chain storage to reduce on-chain burden, i.e., storing file hashes on-chain and content in IPFS, yet they typically focus on general data storage and retrieval rather than DID-specific, task-level decision making under device constraints [10], [11]. Another line of work improves blockchain scalability via Merkle-tree–based on-chain/off-chain storage, thereby alleviating the storage pressure of DID management [8], [9], [12]. However, Merkle tree-based commitment scheme has expensive costs of proof aggregation, nor does it enable storage decisions that are driven by DID holders' constraints in heterogeneous node environments. Meanwhile, preference-driven scheduling has been widely studied in cloud/edge workflow scheduling [13], [14], [15], including two-stage preference-aware optimization. Nevertheless, these methods typically do not incorporate encryption schemes, do not address the privacy requirements of DID data, and do not model the coupling between on-chain and off-chain storage.

To address the issues above, in this paper we propose a *Scalable On-Chain-Off-Chain Storage for Decentralized Identity* (SCOOP) scheme that aims at optimizing the data storage in blockchain-based DID for CPSS. It involves the identity verification and authentication for both human users and physical equipment/devices. Fig. 1 illustrates a high level architecture of our proposed scheme for CPSS-oriented DID

- *K. Ding, T. Xie, K. Gai, S. Wang, J. Gao, and L. Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China, 100081, {3220185089, 3120215672, gaikeke, 3220215214, 3120235247, liehuangz}@bit.edu.cn. K. Gai is also with the School of Artificial Intelligence, Beijing Institute of Technology, Beijing, China, and Zhongguancun Academy, Beijing, China.*
- *J. Yu is with Key Laboratory of Ethnic Language Intelligent Analysis and Security Governance of MOE, Minzu University of China; J. Yu is also with the School of Information Engineering, Minzu University of China. Email: jing.yu@muc.edu.cn.*
- *W. Meng is with School of Computing and Communications, Lancaster University, United Kingdom. Email: weizhi.meng@ieee.org.*
- *This work is supported by the National Natural Science Foundation of China (Grant Nos. U24B20146, 62372044) and Beijing Nova Program (Grant No. 20250484921).*
- *T. Xie is a co-first author (3120215672@bit.edu.cn), Keke Gai is the corresponding author (gaikeke@bit.edu.cn). Co-corresponding author: W. Meng (weizhi.meng@ieee.org)*

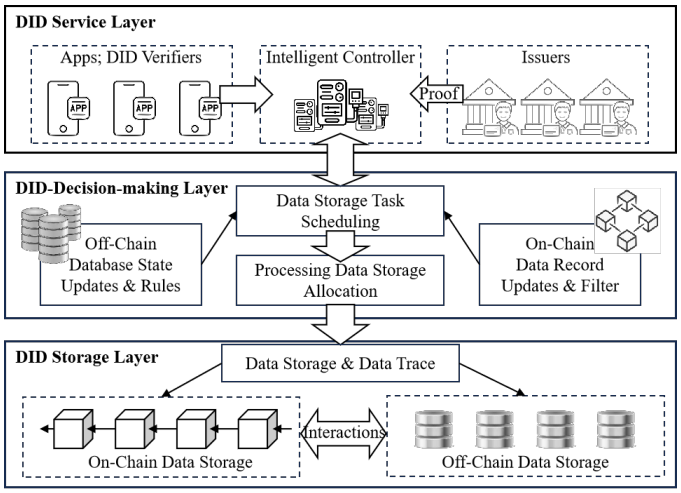


Fig. 1: High level architecture of the proposed scheme.

To be specific, the service layer is an interface for DID service offerings, in which the service requests collections, e.g., forwarding or filtering, are all handled by an *Intelligent Controllers* (IC) that is responsible for identifying/classifying DID service requests. DID-decision-making layer processes a storage task scheduling from making an optimal on-chain-off-chain strategy. In the storage layer, it is a physical object space that covers a few tasks, which include completing data storage on-chain/off-chain, data interactions between on-chain data and off-chain data, and offering traceability.

The strategy of data storage is deemed to be a decision-making task in our scheme. We propose SCOOP, a two-stage on chain and off chain decision mechanism that balances storage savings and efficiency, where different DID applications can adopt different storage strategies. First, *Optimal Storage Task Scheduling* (OSTS) schedules DID storage tasks across heterogeneous storage nodes and minimizes the overall storage cost under latency constraints, thereby adapting to the heterogeneity of CPSS environments. Then, because different cryptographic choices such as Pedersen, SHA 256, and Poseidon incur different storage overheads such as key sizes and proof sizes, *Optimal Storage Strategy* (OSS) determines a fine grained encryption centric storage strategy for each DID Holder within a limited latency budget, so as to satisfy security and DID privacy requirements while reducing off chain storage costs. Furthermore, we design a multilinear tree based commitment scheme to reduce the on chain storage cost of DID identity data. Compared with Merkle tree based approaches, our commitment scheme supports sublinear time aggregation of commitment proofs and enables efficient proof updates. Main contributions of this paper are threefold:

1) SCOOP proposes a two stage storage mechanism that enables scalable DID storage across heterogeneous on chain and off chain media. By decoupling storage task scheduling for heterogeneous CPSS nodes from fine grained, encryption oriented storage strategy selection, our approach dynamically assigns DID storage tasks under latency constraints and minimizes overall DID storage cost while preserving required security guarantees.
2) Our OSS explicitly accounts for the heterogeneous overhead introduced by cryptographic choices, including both storage overhead and execution time. Specifically, SCOOP treats cryptographic operations as decision variables and optimizes them under latency constraints. In contrast to cloud edge storage schemes that adopt a single fixed encryption strategy, SCOOP improves the security efficiency trade off and reducing off-chain storage overhead induced by cryptographic artifacts (such as proof and key materials).
3) We introduce a bilinear tree structured verifiable commitment scheme for the DID registration stage. It supports sublinear time overhead of proof aggregation and efficient proof updates. Compared with Merkle tree-based on-chain-off-chain commitments, our commitment scheme reduces on-chain storage cost and lowers the consumption for maintaining and updating proofs.

The reminder of this paper is organized as follows. The model design of SCOOP is given in Section 2. In Section 3, we give detailed descriptions of the multilieaner tree-based commitment scheme, the OSTS algorithm and the OMS algorithm. Section 4 shows the security analysis of our verifiable DID registration.s In addition, Section 5 presents partial experimental results to verified the proposed model SCOOP. Section 6 reviews the latest work related to DID. Finally, in Section 7, we draw conclusions of this work.

## 2 Proposed Model

### 2.1 Design Goals

Our model mainly covers two design goals as follows:

- **Verifiable DID Management.** Our blockchain-based DID registration, update, and aggregation processes are fully verifiable. Given that CPSS environments involve high-frequency DID updates and large-scale identity data storage, on-chain commitments must support efficient aggregation and maintenance. Specifically, both the update and aggregation overheads of DID commitments grow sublinearly with respect to the number of registered identities, which ensures scalability and maintainability of the DID management.
- **Efficient DID On-chain-off-chain Storage.** We model DID on-chain and off-chain storage as a two-stage storage optimization problem. In each stage, storage decisions are made by minimizing storage cost under time constraints, which enables timely identity verification and data access in CPSS. Moreover, the second-stage storage strategy explicitly accounts for the additional storage overhead introduced by different cryptographic primitives, allowing the DID on-chain-off-chain scheme to balance security guarantees and storage efficiency in a principled manner.

Moreover, the second design goal for efficient on-chain-off-chain storage involves two objectives. On one hand, our model targets at generating an optimal task allocation

TABLE 1: Notations and Descriptions

| Symbol | Description |
|---|---|
| OnSN | On-chain storage node |
| OffSN | Off-Chain storage node |
| $\mathcal{S}$ | Set of storage tasks, $S = \{t_1, t_2, \ldots, t_N\}$. |
| $t_i$ | The $i$-th storage task. |
| $N$ | Total number of storage tasks. |
| $\mathcal{D}$ | Set of OnSNs and OffSNs. |
| $n_j$ | The $j$-th storage node. |
| $\phi(t_i)$ | Storage node assigned to task $t_i$. |
| $E(t_i, n_j)$ | Storage consumption of executing task $t_i$ on node $n_j$. |
| $M_{ij}$ | Data transfer cost between storage nodes $n_i$ and $n_j$. |
| $T_c$ | Global latency constraint. |
| $\mathcal{MP}$ | Set of candidate storage plans for DID holders. |
| $p_i$ | The $i$-th storage plan. |
| $\psi(p_i)$ | Storage node selected to execute plan $p_i$. |
| $C(p_i, \psi(p_i))$ | Storage cost of executing plan $p_i$. |
| $T(p_i, \psi(p_i))$ | Execution latency of plan $p_i$. |
| $\mathcal{TBN}$ | Storage cost table of OnSNs/OffSNs. |
| $TBed$ | Task storage cost demand table. |
| $TBem$ | Storage cost mapping table derived from $\mathcal{TBN}$ and $TBed$. |
| $TST$ | Intermediate table generated by the TSTC operation. |
| $TBot$ | Optimal task storage allocation plan. |
| $Y$ | Sub-optimal task allocation solution. |
| $\lambda$ | Security parameter. |
| $\mathbf{X}$ | Identity attribute vector, $\mathbf{X} \in \mathbb{Z}_p^l$. |
| $\mathcal{C}$ | Commitment of identity attributes. |
| $\pi$ | the proof of the corresponding commitment. |
| $sk, vk$ | Proving key and verification key. |

strategy for on-chain/off chain DID storage to minimize the total storage consumption for completing DID-related services, under a constraint of data volume size (i.e., total volume of data stored on-chain and off-chain). We remark that on-chain storage costs primarily involve transaction fees for smart contracts; off-chain storage costs primarily involve bandwidth and data transmission costs. The other goal is to construct a traceable DID mechanism that supports efficient data acquisition and interactions between on-chain and off-chain data storage. The main issues addressed by our SCOOP model are twofold, including (i) making task scheduling plan in a heterogeneous on-chain node and off-chain node environment and (ii) making optimal storage therapy in a heterogeneous node resource environment. The notations and the corresponding descriptions is shown in Table 1. We formally clarified the problem statement in Definitions 2.1 and 2.2.

**Definition 2.1** (Storage Resource Allocation Problem)**.** ***Input:*** *A set $\mathcal{S}$ composed of storage tasks $\{t_1, t_2, \ldots, t_N\}$ (where $N$ is the number of tasks), and a set $\mathcal{D}$ composed of heterogeneous storage nodes $\{n_1, n_2, \ldots, n_D\}$ (where $D$ is the number of nodes and $N \leq D$). The input also includes a cost table $\mathbf{T}$ that records the storage cost and computing time consumption for executing tasks on different nodes, as well as the data transfer cost $M_{ij}$ between any two nodes $n_i$ and $n_j$.* ***Output:*** *The mapping table $\mathcal{T}_{map}$ of the approximate optimal task scheduling plan. The objective is to minimize the total storage consumption under computing time constraints:*

$$E_{total} = \min \sum_{i=1}^{N} E(t_i, \phi(t_i)), \tag{1}$$

*where $t_i \in \mathcal{S}$ represents a storage task, $\phi(t_i) \in \mathcal{D}$ denotes the storage node assigned to task $t_i$, and $E(t_i, \phi(t_i))$ denotes the storage consumption incurred when executing $t_i$ on node $\phi(t_i)$.*

We note that the storage cost on on-chain and off-chain nodes depends on the adopted cryptographic primitives (e.g., hash functions and commitment schemes), which introduce additional storage overhead.

**Definition 2.2** (Cryptographic Primitive Cost Optimization under Time Constraints)**.** ***Input:*** *A set of DID storage plans $\mathcal{MP} = \{p_1, p_2, \ldots, p_N\}$, where each plan specifies a feasible combination of storage operations, and a data transfer cost table $\mathbf{M}$ representing the communication cost between any two storage nodes $R_i$ and $R_j$. Each plan is associated with an execution latency function $T(p_i, \psi(p_i))$.* ***Output:*** *An approximate optimal storage strategy table $\mathcal{OST}$. The objective is to minimize the total cryptographic primitive-related storage cost under a given time constraint $\mathcal{T}_c$:*

$$C_{total} = \min \sum_{i=1}^{N} C(p_i, \psi(p_i)), \quad s.t. \sum_{i=1}^{N} T(p_i, \psi(p_i)) \leq \mathcal{T}_c, \tag{2}$$

*where $p_i \in \mathcal{MP}$ denotes a storage plan, $\psi(p_i)$ denotes the storage node selected for executing $p_i$, $C(p_i, \psi(p_i))$ denotes the corresponding storage and communication cost, and $T(p_i, \psi(p_i))$ denotes the execution latency.*

We note that the global optimal solution for the objective functions defined in Eq. (1) and Eq. (2) is computationally intensive, especially as the number of tasks ($N$) and nodes ($D$) increases. This resource allocation problem is essentially NP-hard [16], [17], [18].

## 2.2 Model Design

### 2.2.1 System Model

The technical structure of our SCOOP model is shown in Fig. 2. The user side of the DID service layer in the model mainly includes three types of service objects, namely the DID Holder, DID Issuer, and DID Verifier. In addition, the DID decision-making layer and the DID storage layer consist of three key components: the *Intelligent Controller* (IC), the *On-Chain Storage Node* (OnSN), and the *Off-Chain Storage Node* (OffSN). The role of the IC component is to receive and record identity data from DID Issuers and DID Verifiers (including authentication information from the DID Holder and verifiable credential data), filter and categorize DID data, and facilitate data transmission between the DID application layer and the DID decision-making layer. Specifically, OnSN and OffSN are heterogeneous secure storage media that employ different encryption algorithms, storage methods, and privacy computation schemes. Therefore, the DID decision-making layer of SCOOP generates corresponding DID storage task scheduling strategies based on the secure storage requirements of the DID Holder, flexibly allocating OnSNs and OffSNs on-chain and off-chain to achieve optimal secure DID data storage at low cost. Subsequently, based on the results of the DID storage scheduling, the DID data storage tasks are forwarded to the OnSNs and OffSNs for computation. Furthermore, the OnSNs and OffSNs further compute the optimal DID data storage strategy through the OSS algorithm to provide users

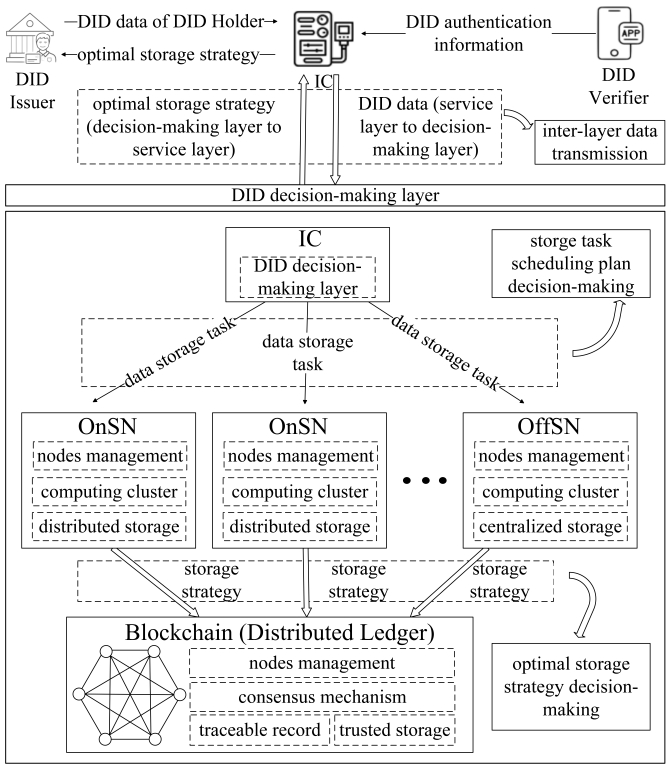


Fig. 2: The technical structure of SCOOP.

with the best strategy, while recording the DID storage strategy in the blockchain distributed ledger for traceability.

The SCOOP model proposed in this paper mainly includes five phases, namely the DID registration phase, the data collection phase, the data storage task scheduling decision-making phase, the optimal storage plan decision-making phase and the distributed ledger storage stage. Fig. 3 shows the high-level workflow for SCOOP.

**Phase I (DID registration phase)**: During the DID registration phase, a user's identity attributes are securely bound to a verifiable registration. In SCOOP, identity data are modeled as a structured vector and associated with the registration via a vector commitment construction. Specifically, each user encodes identity attributes into a vector $\mathbf{X} = (x_0, x_1, \ldots, x_{l-1}) \in \mathbb{Z}_p^l$ and generates a corresponding commitment together with a validity proof. In Step (1), the physical condition and other related attributes of the DID Holder are uploaded to the IC by DID Issuers.

The proposed vector commitment scheme consists of the following five functions:

- $(pp, sk, vk) \leftarrow \mathsf{Gen}(\lambda)$: Given a security parameter ($\lambda$), this algorithm outputs the public system parameters ($pp$), a secret key ($sk$), and a public verification key ($vk$).
- $(\mathcal{C}, \pi_u) \leftarrow \mathsf{Commit}(\mathbf{X}, x_u, u)$: This algorithm inputs an vector ($\mathbf{X}$), the element ($x_u$) with the specific position ($u$), and outputs a commitment ($\mathcal{C}$) of vector ($\mathbf{X}$) and the proof ($\pi_u$) of element ($x_u$).
- $\pi_{\mathcal{U}}, \mathcal{C}_{\mathcal{U}} \leftarrow \mathsf{Aggregate}(x_u, \pi_u, \mathcal{U})$ $(u \in \mathcal{U})$: Given an index set ($\mathcal{U}$) specifying selected vector positions, together with the corresponding elements ($x_u$) and proofs ($\pi_u$), this algorithm outputs an aggregated proof $\pi_{\mathcal{U}}$ and an aggregated commitment $\mathcal{C}_{\mathcal{U}}$.
- $b \leftarrow \mathsf{Verify}(\mathcal{C}, \mathcal{U}, \pi_{\mathcal{U}}, (x_u)_{u\in\mathcal{U}}, (vk_u)_{u\in\mathcal{U}})$: This algorithm verifies whether the aggregated proof $\sigma_{\mathcal{U}}$ correctly attests that the commitment $\mathcal{C}$ contains the specified vector elements $(x_u)_{u\in\mathcal{U}}$. It outputs $b \in \{0, 1\}$ to indicate acceptance or rejection.
- $(\mathcal{C}', \pi') \leftarrow \mathsf{Update}(\mathcal{C}, j, \alpha)$: Given an existing commitment ($\mathcal{C}$), an position ($j$), and the update values ($\alpha$) that specifies the transformation applied to position ($j$), this algorithm outputs an updated commitment ($\mathcal{C}'$) and a corresponding proof ($\pi'$) reflecting the modification.

**Phase II (Data collection phase)**: This phase is responsible for the collection and preliminary processing of important DID information. Steps (2) to (4) describe the process of the IC collecting DID data and related OnSN/OffSN information. In Step (2), the DID Verifiers upload multiple DID authentication records to the IC. In Step (3), the data collected by the IC are filtered and processed. The filtered DID data storage tasks are transmitted from the DID application layer to the DID decision-making layer. In Step (4), the IC pre-collects quantitative information on the computing capabilities of OnSNs and OffSNs, and stores the computing time and storage consumption information of OnSNs/OffSNs for specific tasks as an OnSN/OffSN cost table ($\mathcal{TBN}$).

**Phase III (Data storage task scheduling decision-making phase)**: This phase describes the process by which the IC determines a task scheduling plan based on the stored $\mathcal{TBN}$ and the workload levels of the data storage tasks. The IC receives a series of data processing tasks from DID Issuers and DID Verifiers and estimates the workload level of each task according to the basic information in the task header. The SCOOP model defines the input tasks as a set. The task scheduling plan generated by the IC must satisfy the computing time constraint requirement $TC$. The IC first constructs a preliminary task scheduling plan using a greedy algorithm with time priority and then estimates the total computing time cost required to complete the task computation. The storage consumption of the OnSNs/OffSNs is not considered in this process. The generated preliminary task scheduling plan can be regarded as a sub-optimal task allocation solution that satisfies the computing time constraint.

Next, the IC uses the preliminary task scheduling plan to generate a new storage-cost-oriented $\mathcal{TBN}$. The IC takes this $\mathcal{TBN}$ as input and computes the storage cost of all OnSNs and OffSNs. Then, the IC selects the task scheduling scheme with the minimum storage cost through the *Task Standard Table Creation* (TSTC) operation and the *Task Transfer* (TT) operation, while ensuring the computing time constraint. At this point, the task scheduling plan is optimal. In the final step, Step (5), the OSTS algorithm outputs a set of OnSN/OffSN addresses, and the IC forwards the data processing tasks to the corresponding OnSNs and OffSNs.

**Phase IV (Optimal storage plan decision-making phase)**: In this phase, the OnSNs and OffSNs compute the optimal storage strategy based on the information contained in the data processing task body. Using the OSS algorithm, each storage node derives an optimal storage strategy that

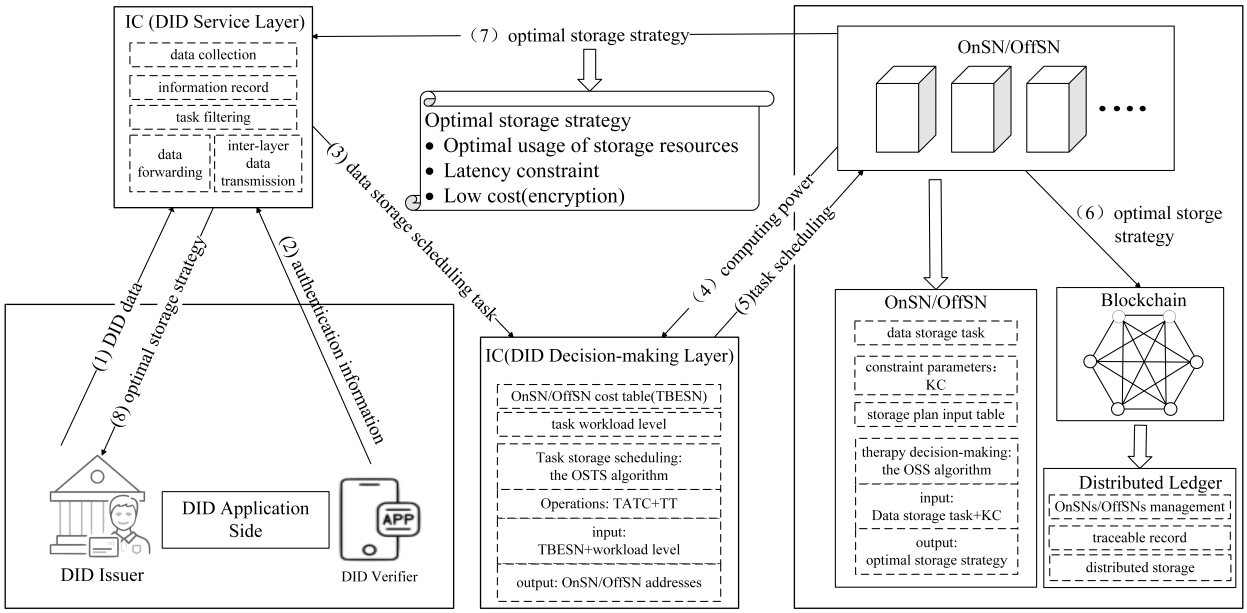


Fig. 3: High-level workflow for SCOOP.

minimizes storage cost under the given latency constraint. Similar to the task scheduling decision-making phase, the OnSNs and OffSNs first map the relevant information in the task body to construct an initial cost table. Subsequently, the data categories are refined to generate a cryptography-oriented derivation table, which serves as a new input for determining the optimal strategy through dynamic programming and iterative optimization. After the computation is completed, the resulting storage strategy is returned to the IC as a response to the task scheduling decision, thereby enabling data synchronization between the IC and the OnSNs/OffSNs. At this point, the computation cycle of the DID data processing task is completed.

**Phase V (Distributed ledger storage phase)**: This phase focuses on the distributed storage of the optimal storage strategy associated with each user. By leveraging a distributed ledger, the users' storage strategies become traceable, thereby serving a role analogous to an electronic record. In Step (6), acting as nodes of the distributed ledger, the OnSNs and OffSNs package and store the generated optimal storage strategies together in the ledger. In Step (7), the OnSNs and OffSNs transmit the optimal storage strategy to the IC. In Step (8), the IC forwards the storage strategy to the corresponding users.

### 2.2.2 On-chain-off-chain Storage Optimize

SCOOP involves two decision-making processes: (i) task scheduling across OnSNs and OffSNs, and (ii) storage strategy optimization based on DID storage plan information.

The first process addresses the task scheduling problem and is executed by the IC. Due to heterogeneous hardware and network configurations, OnSNs and OffSNs exhibit different computing capabilities, leading to varying storage and computing time costs for the same task. The IC evaluates task workload levels and node capabilities, and organizes the corresponding storage and computing time costs into an OnSN/OffSN cost table $\mathcal{TBN}$. Based on $\mathcal{TBN}$, tasks

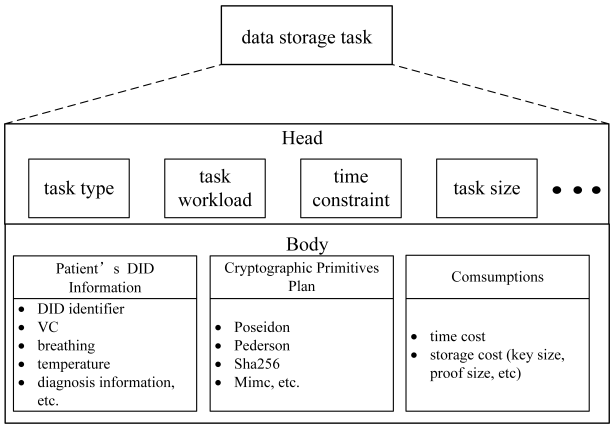


Fig. 4: The structure of data storage tasks for the motivation example.

are assigned to appropriate OnSNs and OffSNs by minimizing storage consumption under computing time constraints, yielding an optimal task allocation plan. Each entry in $\mathcal{TBN}$ is represented as a five-tuple $\langle i, t, E, T_i, MC_{ij} \rangle$, where $i$ denotes the node identifier, $t$ the task type, $E$ and $T_i$ the storage and computing time costs of executing task $t$ on CPSS node $i$, and $MC_{ij}$ the transfer cost between nodes. The table $\mathcal{TBN}$ serves as the input to the OSTS algorithm, whose core operations are the TSTC and TT. The second process focuses on storage strategy optimization based on different costs of cryptographic primitives. After receiving a storage task, an OnSN or OffSN processes and classifies the task data, and applies the OSS algorithm using task attributes and constraint parameters (e.g., computing time and storage cost) as inputs. The OSS algorithm outputs an optimal storage strategy specifying suitable storage nodes and resources for the DID Holder.

**Motivation Example.** In a smart healthcare environment, a patient's DID is frequently used to support identity ver-

ification, medical data access, and cross-institutional collaboration. Each medical service request is abstracted as a data storage task with a structured head and body, as illustrated in Fig. 4. The task head contains metadata such as task type, workload level, time constraint, and task size, which reflect the urgency and scale of the medical operation. For example, emergency diagnosis requests impose strict time constraints, while routine health monitoring tasks are less time-sensitive. Based on the task head information, the OSTS algorithm assigns each storage task to an appropriate OnSN or OffSN by minimizing storage consumption under the given time constraints, thereby ensuring timely medical services while reducing system overhead.

After task scheduling is completed, the selected OnSN or OffSN determines the optimal storage strategy using the task body information. The task body includes the patient's DID-related attributes (e.g., identifier and physiological indicators), candidate cryptographic primitives, and the associated resource consumptions. In a medical setting, different cryptographic choices introduce distinct storage and processing overheads, such as key sizes and proof sizes, which directly affect the efficiency of on-chain and off-chain storage. The OSS algorithm takes these factors as inputs and computes an optimal storage strategy that balances security requirements and on-chain/off-chain storage efficiency under time constraints. As a result, sensitive medical identity data are stored securely and efficiently, while supporting continuous updates and traceable management.

## 3 ALGORITHMS

For verifiable DID registration, we adopt a multilinear tree-based commitment scheme to support effective proof updates and aggregation. In order to support the IC to compute the optimal task storage scheduling scheme of the OnSN/OffSN, we propose an OSTS algorithm. In order to support the OnSN/OffSN to compute the optimal storage strategy plan, we propose an OSS algorithm. The key of the proposed algorithm is to transform the sub-optimal solution into the optimal solution, which solves the problem of the optimal solution that the greedy algorithm cannot provide. In the following subsections, we introduce the design and implementation process of the algorithms in detail.

### 3.1 Verifiable DID registration

After completing the registration request, SCOOP generates cryptographic materials. The DID Verifier validates the correctness of the submitted commitment by checking the associated proof. Once the verification succeeds, the commitment-related information is persistently stored in the distributed ledger. In large-scale CPSS scenarios, identity data are frequently updated, thus SCOOP employs a vector commitment scheme built upon a multilinear tree structure, which enables effective aggregation and update operations with sublinear complexity.

**Commitment and Proof Generation.** Inspired by Hyperproof [19] and polynomial commitment schemes [20], the identity information vector $\mathbf{X} = (x_0, x_1, \ldots, x_{m-1}) \in \mathbb{Z}_p^m$ (with $m = 2^l$) is first represented as a multilinear extended polynomial ($f$). The commitment is then instantiated within a multilinear pairing, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups generated by $\mathrm{g}_1$ and $\mathrm{g}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ denotes the multilinear pairing. For each vector position, the verification key ($vk$) corresponding to index $u$ is defined as $\mathrm{g}_2^{t_j - u_j}$ for $j \in [0, l]$. The resulting commitment is expressed in Eq. (3).

$$\mathcal{C} = \mathrm{g}_1^{f(\mathbf{A})} = \mathrm{g}_1^{\sum_{k=0}^{m-1} x_k \cdot \Gamma_{k,l}(\mathbf{A})} = \prod_{k=0}^{m-1} \left(\mathrm{g}_1^{\Gamma_{k,l}(\mathbf{A})}\right)^{x_k}, \quad (3)$$

Here, $\mathbf{A}$ denotes a trapdoor, and $\Gamma_{k,l}$ represents the selection function. Specifically, for $\mathbf{A} = (a_l, a_{l-1}, ..., a_1) \in_R \mathbb{Z}_p^l$, we define the selection functions $\Lambda_{k_j}$ and $\Gamma_{k,l}$ ($k \in [0, 2^l)$) as follows:

$$\Lambda_{k_j}(a_j) = \begin{cases} a_j, & k_j = 1 \\ 1 - a_j, & k_j = 0 \end{cases}, \quad \Gamma_{k,l}(\mathbf{a}) = \prod_{j=1}^{l} \Lambda_{k_j}(a_j) \quad (4)$$

We note that $\Gamma_{0,0}(\mathbf{A}) = 1$. The trapdoor $\mathbf{A}$ guarantees the binding and hiding properties of the commitment $\mathcal{C}$. To improve the efficiency of proof generation, a binary linear tree structure is introduced to recursively decompose the multilinear polynomial. All commitment components located on the path from the leaf associated with $x_u$ to the root collectively form the proof $\pi_u$ for element $x_u$.

**Proof Aggregation.** Within the SCOOP, individual proofs corresponding to elements $x_u$ of the identity vector $\mathbf{X}$ can be combined into a single aggregated proof. This aggregated proof serves to jointly attest to the correctness of the entire identity vector. Motivated by Hyperproof [19], we adopt the *Non-interactive Inner Product Arguments* (IPA) to realize proof aggregation. Specifically, the IPA construction produces an aggregated proof $\pi_{\mathcal{U}}$ together with the corresponding aggregated commitment $\mathcal{C}_{\mathcal{U}}$ from the set of individual proofs and their verification keys ($vk$).

**Proof Verification.** During verification, the public verification key is $\mathrm{g}_2$. The prover submits the proof $\pi_u = (\chi_{u,l}, \chi_{u,l-1}, \ldots, \chi_{u,1})$, where each $\chi_{u,j}$ corresponds to a node on the $u$-th path of the binary linear tree. The DID verifier evaluates the proof by checking the pairing equation $e(\mathcal{C}/\mathrm{g}_1^{a_u}, \mathrm{g}_2)$ against $\prod_{j\in[0,l]} e(\chi_{u,j}, \mathrm{g}_2^{a_j - u_j})$. The proof is accepted if Eq. (5) is satisfied.

$$e(\mathcal{C}/\mathrm{g}_1^{a_u}, \mathrm{g}_2) = \prod_{j\in[0,l]} e(\chi_{u,j}, \mathrm{g}_2^{a_j - u_j}). \quad (5)$$

**Commitment and Proof Update.** To support effective identity management in SCOOP, the commitment scheme allows efficient updates when the position of an element in the identity vector changes. Assume that the position of element $x_u$ in $\mathbf{X}$ is shifted by $\alpha$. Updating the associated commitment and proof requires auxiliary information defined in Eq. (6).

$$\beta_{u'} = \{\beta_{u',j} \mid j \in [0, l]\} = \left\{\mathrm{g}_1^{\Gamma_{u',j}(\mathbf{A})} \mid j \in [0, l]\right\}. \quad (6)$$

The updated proof elements are computed as

$$\chi'_{u',j} = \chi_{u',j} \cdot (\beta_{u',j-1})^{\alpha} = \chi_{u',j} \cdot \left(\mathrm{g}_1^{\Gamma_{u',j-1}(\mathbf{A})}\right)^{\alpha}. \quad (7)$$

If indices $u$ and $u'$ share $d$ identical bits, then for any $j \in [l-d+1, l]$, it holds that $u_j = u'_j$. Accordingly, the updated proof $\pi_u$ is computed as

$$\chi'_{u,j} = \chi_{u,j} \cdot (\beta_{u,j-1})^{\alpha} = \chi_{u,j} \cdot \left(\mathrm{g}_1^{\Gamma_{u,j-1}(\mathbf{A})}\right)^{\alpha}. \quad (8)$$

Finally, the commitment is updated as

$$\mathcal{C}' = \mathcal{C} \cdot \mathrm{g}_1^{\Gamma_{u',l}(\mathbf{A})} = \mathcal{C} \cdot (\beta_{u',l})^{\alpha}. \quad (9)$$

### 3.2 Optimal Storage Task Scheduling (OSTS) Algorithm

The purpose of the OSTS algorithm is to generate an encryption-optimized storage task scheduling scheme under time constraints. The algorithm consists of two main processes.

The first process aims to obtain a sub-optimal task scheduling scheme. A greedy algorithm is adopted as one feasible approach to derive such a sub-optimal solution. The second process focuses on transforming the sub-optimal scheduling scheme into the optimal task scheduling scheme. This transformation constitutes the core of the OSTS algorithm. We further divide the second process into two execution steps as follows: (i) generating an intermediate table based on the storage cost table of the OnSN/OffSN and the sub-optimal task scheduling scheme; (ii) generating the optimal storage task scheduling scheme based on the intermediate table and the sub-optimal task scheduling scheme.

The pseudocode of the OSTS algorithm is provided in Alg. 1. The input of the algorithm includes the storage cost table $\mathcal{TBN}$ and the task storage cost demand table TBed, while the output is the optimal task storage allocation plan TBot. The key steps of the algorithm are summarized as follows:

1) Initialization: initialize the optimal task scheduling table TBot.
2) Sub-optimal scheduling: generate a storage cost mapping table TBem and apply a greedy algorithm to obtain a sub-optimal scheduling scheme $Y$ based on the cost table $\mathcal{TBN}$ and the task storage cost demand table TBed.
3) Intermediate table generation: take the sub-optimal scheduling scheme $\mathcal{Y}$ and the cost mapping table TBem as inputs to the TSTC operation, and execute the operation to obtain the intermediate table $\mathcal{TST}$.
4) Optimal scheduling transformation: use the intermediate table $\mathcal{TST}$ and the sub-optimal scheduling scheme $\mathcal{Y}$ as inputs to the TT operation, and execute the operation to generate the optimal task scheduling scheme TBot.
5) Output: return the optimal task scheduling scheme TBot.

#### 3.2.1 Task Standard Table Creation (TSTC) Operation Algorithm

The purpose of the TSTC operation algorithm is to generate an intermediate table that reflects the potential existence of an optimal scheduling scheme. The core idea of the TSTC algorithm is to subtract the storage consumption cost of the selected computing unit from the storage consumption costs of the same input task across all available computing units in the storage cost consumption mapping table TBem.

The pseudocode of the TSTC algorithm is presented in Alg. 2. The TSTC algorithm takes two inputs: the storage cost consumption mapping table TBem and the sub-optimal task allocation solution $Y$. The storage cost mapping table TBem is a two-dimensional matrix, where TBem[$i$][$j$] denotes the storage consumption required for the $i$-th input task to be processed by the $j$-th edge server. The sub-optimal solution $Y$ is a one-dimensional array, and $Y(i)$ indicates the index of the edge server assigned to the $i$-th task. The output of the algorithm is an intermediate table TST in matrix form. The key steps of the TSTC algorithm are summarized as follows:

1) Input the cost mapping table TBem and the sub-optimal task allocation plan $Y$, and initialize the output table TST.
2) Execute a FOR loop to read the corresponding elements in $Y$, and subtract the storage consumption cost of the selected edge server from all corresponding elements in TBem to obtain the elements of the table TST. The FOR loop terminates after all elements in the TBem table have been processed.
3) Return the intermediate table TST.

**Algorithm 1** Optimal Storage Task Scheduling Algorithm

**Require:** $\mathcal{TBN}$, Storage cost demand table(TBed);
**Ensure:** TBot;
1: Initialize the optimal solution TBot;
2: Generate table TBem and sub-optimal scheduling scheme $Y$;
3: Perform TSTC (Alg.2) standardized sub-optimal solution to obtain the intermediate table TST;
4: Perform TT (Alg.3) to get the optimal task scheduling TBot;
5: **return** TBot;

**Algorithm 2** Task Standard Table Creation Algorithm

**Require:** TBem, $Y$;
**Ensure:** TST;
1: Input the table TBem and sub-optimal task scheduling plan $Y$;
2: Create table TST;
3: Perform TSTC (1) standardized sub-optimal solution to obtain the intermediate table TST;
4: Perform TT (Algorithm 3) to get the optimal task scheduling TBot;
5: **for** $i$ in TBem **do** {/*Traverse each row of TBem, i represents the row number*/}
6: $k$ = $Y[i]$;
7: **for** $j$ in TBem **do** {/* Traverse each column of TBem, j represents the column number */}
8: TST[$i$][$j$] = TBem[$i$][$j$] – TBem[$i$][$k$];
9: **end for**
10: **end for**
11: **return** TST;

#### 3.2.2 Task transfer (TT) Algorithm

The Task Transfer (TT) Algorithm is a core algorithm designed to reduce the total storage consumption by transforming the sub-optimal solution obtained from the greedy algorithm through the selection of edge computing units, thereby deriving an approximate optimal solution with minimal total storage consumption. Since the greedy algorithm generates a task allocation plan based on task priority, a situation may arise in which, for the same computing task, the selected computing unit consumes more storage resources than other unselected computing units that are already occupied by different tasks. Therefore, the objective of the TT algorithm is to perform transfer-based transformations among tasks and ultimately obtain a task allocation plan with the minimum total storage consumption.

The pseudocode of the TT algorithm is provided in Alg. 3. The TT algorithm takes two input parameters: the intermediate table TST in the form of a two-dimensional matrix and the sub-optimal task allocation plan $Y$ in the form of a one-dimensional array. Here, $Y(i)$ denotes the index of the edge server assigned to the $i$-th task. The output of the algorithm is the optimal task allocation plan TBot, which is also represented as a one-dimensional array.

The core idea of the TT algorithm is to reduce the total storage consumption by exchanging the allocations of two tasks at a time. Each transfer operation only involves the two selected tasks and is independent of all other tasks. Consequently, the original multi-row matrix problem can be reduced to a two-row matrix problem. Similarly, since only two edge computing units participate in each transfer operation, the multi-column matrix problem can be simplified into a two-column matrix problem. As a result, the task transfer process can be further formulated as a $2 \times 2$ matrix optimization problem, which significantly reduces the complexity of the solution. The key steps of the TT algorithm are summarized as follows:

1) Initialization: input the intermediate table TST and the sub-optimal solution $Y$, and initialize the optimal allocation table TBot.
2) Task transfer operation: execute FOR loops to traverse all elements in the TST table. When an element with a value less than zero is encountered, the TST table is searched according to predefined rules. If a feasible task exchange exists, the corresponding two tasks are selected for transfer, the sub-optimal solution $Y$ is updated accordingly, and the TST table is updated based on the transfer results. The process continues until the FOR loop terminates.
3) Output: output the optimal task allocation solution TBot.

**Algorithm 3** Task Transfer Algorithm

**Require:** TST, Y;
**Ensure:** TBot;
Input the intermediate table TST and the sub-optimal task scheduling plan $Y$;
Initialize table TBot;
**for** $i$ in TST **do** {/*Traverse each row of TST, $i$ represents the row number*/}
  **for** $j$ in TST **do** {/* Traverse each column of TST, $j$ represents the column number */}
    **if** $TST[i][j] < 0$ **then**
      **for** $k$ in TST **do** {/* Traverse each column of TST, $j$ represents the column number */}
        **if** $TST[i][k] = 0$ **then**
          break;
        **end if**
      **end for**
      $min \leftarrow 0$;
      $changeIndex \leftarrow -1$;
      **for** $l$ in TST **do** {/* Traverse each row of TST, $l$ represents the row number */}
        **if** $TST[l][k] = 0$ and $min > TST[i][j] + TST[l][k]$ **then**
          $min = TST[i][j] + TST[l][k]$;
          $changeIndex \leftarrow l$;
        **end if**
      **end for**
      **if** $changeIndex! = 0$ **then**
        $Y[i] \leftrightarrow Y[changeIndex]$;
        Each element of the $changeIndex$-th row of TST minus $TST[changeIndex][k]$;
        Each element of the $i$-th row of $TST$ add $TST[i][j]$;
      **end if**
    **end if**
  **end for**
**end for**
$TBot \leftarrow Y$;
**return** TBot;

### 3.3 Optimal Storage Strategy (OSS) Algorithm

The primary objective of the OSS algorithm is to derive an optimal storage strategy under explicit latency constraints while preserving data security. Specifically, OSS aims to minimize the storage cost induced by different choices of cryptographic primitives. For vectors of identical size, distinct encryption and commitment schemes introduce heterogeneous overheads, including proof verification time, proof size, encryption key size, and verification key size. Consequently, the adoption of different cryptographic primitives results in a heterogeneous storage environment, which motivates the need for a cost-aware storage optimization algorithm subject to latency constraints.

The pseudocode of the OSS algorithm is presented in Algorithm 4. The input to the OSS algorithm is the storage plan input table $\mathcal{MP}$, and the output is an optimal storage plan for DID Holders. The OSS algorithm consists of four main processes. First, based on the input table $\mathcal{MP}$, the storage cost mapping table MPM is constructed. Second, a greedy algorithm is applied to the cost mapping table MPM to obtain a sub-optimal storage cost strategy, denoted as ST. Third, a subtraction operation is performed on the cost mapping table MPM according to the sub-optimal solution ST. Specifically, for each task, the storage cost selected in ST is subtracted from all corresponding elements in the same row of MPM, resulting in an intermediate table denoted as M-MPM. Finally, task exchange operations are executed based

**Algorithm 4** Optimal Storage Strategy Algorithm

**Require:** MP;
**Ensure:** OST;
1: Generate cost mapping table MPM according to $\mathcal{MP}$ table;
2: Generate the sub-optimal solution $ST$ of the storage cost strategy by executing the greedy algorithm;
3: Initialization the table M-MPM;
4: **for** $i$ in MPM **do** {/*Traverse each row of MPM, $i$ represents the row number*/}
5: $M-MPM[i] \leftarrow MPM[i] - MPM[i][ST[i]]$
6: **end for**
7: **for** $i$ in M-MPM **do** {/*Traverse each row of M-MPM, $i$ represents the row number*/}
8: **for** $j$ in M-MPM **do** {/* Traverse each column of M-MPM, $j$ represents the column number */}
9: **if** $M-MPM[i][j] < 0$ **then**
10: **if** Find a task $k$, after swapping with task $i$, the total cost is reduced **then**
11: $ST[i] \leftrightarrow ST[k]$;
12: /*exchange the value of Y[i] and the value of Y[changeIndex]*/
13: update M-MPM;
14: **end if**
15: **end if**
16: **end for**
17: **end for**
18: $OST \leftarrow ST$;
19: **return** OST;

on the intermediate table M-MPM and the sub-optimal storage strategy ST. Through iterative task transfers, the storage strategy is further optimized, and the final optimal storage strategy OST with the minimum total cost is obtained as the output of the OSS algorithm.

## 4 Security Analysis

Our multilinear tree-based DID register scheme meets the correctness and soundness of the classic commitment scheme. Brief observation are given as follow:

**Theorem 4.1** (**Correctness**). *Let $\mathcal{U} \subseteq \{0, 1, \ldots, m-1\}$ be an index set. If an honest prover $\mathcal{P}$ follows the prescribed protocol and outputs a commitment $\mathcal{C}$ together with an aggregated proof $\pi_{\mathcal{U}}$, then a verifier $\mathcal{V}$, given correct public inputs, always accepts. Formally, the DID registration satisfies* correctness *if*

$$\Pr[\mathsf{Verify}(\mathcal{C}, u, \pi_u, x_u, vk_u) = 1] = 1, \tag{10a}$$
$$\Pr[\mathsf{Verify}(\mathcal{C}, \mathcal{U}, \pi_{\mathcal{U}}, (x_u)_{u\in\mathcal{U}}, (vk_u)_{u\in\mathcal{U}}) = 1] = 1. \tag{10b}$$

*Proof Sketch.* Given a security parameter $\lambda$, public parameters $pp$, and an identity vector $\mathbf{X} = (x_0, x_1, \ldots, x_{m-1}) \in \mathbb{Z}_p^m$, the prover $\mathcal{P}$ executes

$$(\mathcal{C}, \pi_u) \leftarrow \mathsf{Commit}(\mathbf{X}, x_u, u).$$

Because $\mathcal{P}$ is honest, the resulting commitment $\mathcal{C}$ is well-formed and uniquely corresponds to $\mathbf{X}$.

For each $u \in \mathcal{U}$, the prover generates a proof $\pi_u$ certifying that the value $x_u$ is correctly bound to position $u$ in $\mathcal{C}$. All individual proofs are then combined into a single proof

$$\pi_{\mathcal{U}} \leftarrow \mathsf{Aggregate}((x_u, \pi_u)_{u\in\mathcal{U}}).$$

Upon receiving $(\mathcal{C}, \mathcal{U}, \pi_{\mathcal{U}}, (x_u)_{u\in\mathcal{U}}, (vk_u)_{u\in\mathcal{U}})$, the verifier runs

$$b \leftarrow \mathsf{Verify}(\mathcal{C}, \mathcal{U}, \pi_{\mathcal{U}}, (x_u)_{u\in\mathcal{U}}, (vk_u)_{u\in\mathcal{U}}),$$

where $b \in \{0, 1\}$. By construction, the verification algorithm outputs 1 whenever all protocol steps are executed correctly.

**Theorem 4.2** (**Soundness**). *The proposed DID registration satisfies* soundness, *namely that it is computationally infeasible to generate accepting proofs for inconsistent identity values under the same commitment.*

*More precisely, for any two distinct identity vectors $\mathbf{X} \neq \mathbf{X}'$ and any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ outputs a commitment $\mathcal{C}$ together with two accepting verification transcripts that disagree on at least one common index is negligible:*

$$\Pr\left[\begin{array}{c} pp, sk, vk \leftarrow \mathsf{Gen}(\lambda), \\ \left(\mathcal{C}, \begin{array}{c}(\mathcal{U}, x_u, vk_u, \pi_{\mathcal{U}})_{u\in\mathcal{U}}, \\ (\mathcal{V}, x'_v, vk_v, \pi'_{\mathcal{V}})\end{array}\right) \leftarrow \mathcal{A}(\lambda, pp) : \\ \mathsf{Verify}(\mathcal{C}, \mathcal{U}, \pi_{\mathcal{U}}, x_u, vk_u) = 1 \wedge \\ \mathsf{Verify}(\mathcal{C}, \mathcal{V}, \pi'_{\mathcal{V}}, x'_v, vk_v) = 1 \wedge \\ \exists\, t \in \mathcal{U} \cap \mathcal{V} :\ x_t \neq x'_t \end{array}\right] \leq \mathrm{negl}(\lambda). \tag{11}$$

*Proof Sketch.* Assume, towards a contradiction, that there exists an adversary $\mathcal{A}$ that can find two different identity vectors $\mathbf{X}_1 \neq \mathbf{X}_2$ such that

$$\mathsf{Commit}(\mathbf{X}_1) = \mathsf{Commit}(\mathbf{X}_2) = \mathcal{C}.$$

This implies that two distinct exponent representations yield the same group element with respect to the generator $g_1$, which contradicts the $q$-Strong Diffie–Hellman assumption.

Hence, the probability that such a collision occurs is negligible in the security parameter $\lambda$:

$$\Pr[\exists\, \mathbf{X}_1 \neq \mathbf{X}_2 : \mathcal{C} = \mathsf{Commit}(\mathbf{X}_1) = \mathsf{Commit}(\mathbf{X}_2)] \leq \mathrm{negl}(\lambda).$$

By combining correctness and soundness, the proposed multilinear tree-based DID registration provides strong tamper-evidence. The commitment $\mathcal{C}$ uniquely binds the identity vector $\mathbf{X}$, and any unauthorized modification of identity attributes necessarily causes verification failure. Honest registrations are always accepted, whereas forged or altered DID documents are rejected except with negligible probability.

## 5 Experiment and Results

### 5.1 Experimental Configuration

To comprehensively evaluate the performance of the proposed SCOOP model under diverse CPSS scenarios, we conduct a series of simulation-based experiments. All experiments are performed on a workstation equipped with a Windows 10 (64-bit) operating system, an Intel® Core™ i5-1035G1 CPU, and 16 GB of RAM.

In the DID registration phase, the proposed multilinear tree-based commitment scheme is implemented in Golang

using the `mcl` cryptographic library[1]. The implementation is based on the BLS12-381 elliptic curve, which provides efficient arithmetic operations over the algebraic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. These properties make BLS12-381 well suited for pairing-based cryptographic constructions. To further improve efficiency, we incorporate an IPA-based aggregation mechanism derived from the `Hyperproof` framework[2]. This aggregation mechanism enables multiple commitments and proofs to be aggregated and verified within a single verification procedure, thereby reducing computational overhead. The experimental settings for the commitment scheme systematically vary the vector size $w \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$ and the height of the multilinear tree $h \in \{1, 2, 3, 4, 30\}$, in order to evaluate their respective impacts on the performance of the DID registration process.

For the on-chain and off-chain storage task allocation in the SCOOP model, both on-chain and off-chain computations are assumed to be performed by a limited number of storage nodes. The corresponding simulations are conducted using a Java-based simulator, which allows flexible configuration of relevant performance parameters. To emulate heterogeneous DID service scenarios, three experimental settings are adopted. Each experimental setting is executed with $10^3$ repeated iterations to ensure statistical stability and robustness. To better approximate real-world conditions, different probabilistic and structural assumptions are introduced. In the OSTS experiments, storage consumption is modeled using a normal distribution, parameterized by the number of tasks and the types of storage nodes (OnSN/OffSN). In the OSS experiments, we additionally implement Merkle-based Groth16 circuits using four representative cryptographic hash primitives, namely *Poseidon*, *Pedersen*, *MiMC*, and *SHA256*. For each primitive, both the time cost and storage overhead are evaluated under Merkle tree heights $h \in \{1, 2, 3, 4\}$. The OSS experiments are parameterized by the number of tasks and the number of cryptographic primitives employed. Based on the above configurations, three experimental settings are designed as follows:

- **Setting 1**: The number of tasks is fixed as specified in Table 2. This setting aims to investigate the impact of computational resources when the task workload remains constant in combined on-chain and off-chain environments.
- **Setting 2**: The available computational resources are fixed, as shown in Table 2. This setting evaluates how variations in the number of tasks affect system performance under constrained on-chaih=off-chain resources.
- **Setting 3**: Both the number of storage plans and the number of cryptographic primitives are varied, as summarized in Table 2. This setting is designed to assess the cost implications of increasing storage plan complexity and cryptographic diversity under bounded computation-time constraints.

To quantitatively evaluate the performance of OSTS and OSS, we define an evaluation metric $F$, as specified

1. https://github.com/herumi/mcl.git
2. https://github.com/hyperproofs/hyperproofs-go.git

TABLE 2: Experiment Settings

| Settings | Parameters | Number of Parameters |
|---|---|---|
| 1-1 | (tasks, node numbers) | (30,4) |
| 1-2 | (tasks, node numbers) | (30,5) |
| 1-3 | (tasks, node numbers) | (30,6) |
| 1-4 | (tasks, node numbers) | (30,7) |
| 1-5 | (tasks, node numbers) | (30,8) |
| 1-6 | (tasks, node numbers) | (30,9) |
| 2-1 | (tasks, node numbers) | (10,4) |
| 2-2 | (tasks, node numbers) | (20,4) |
| 2-3 | (tasks, node numbers) | (30,4) |
| 2-4 | (tasks, node numbers) | (50,4) |
| 2-5 | (tasks, node numbers) | (100,4) |
| 2-6 | (tasks, node numbers) | (120,3) |
| 3-1 | (tasks, crpto. numbers) | (6,3) |
| 3-2 | (tasks, crpto. numbers) | (6,4) |
| 3-3 | (tasks, crpto. numbers) | (10,4) |

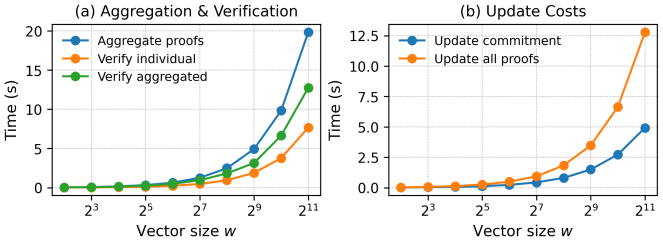


Fig. 5: Time costs of the DID registry under different vector sizes ($h = 30$).

in Eqs. (12) and (13). The metric $F$ is computed as the average value of a binary indicator function over multiple experimental iterations:

$$F = \frac{1}{N} \sum_{i=1}^{N} f(i), \tag{12}$$

$$f(i) = \begin{cases} 1, & \text{if } s[i] = \min\big(s[i], \text{other schemes}[i]\big), \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

where $N$ denotes the total number of experimental iterations. In the $i$-th iteration, $s[i]$ represents the cost incurred by the proposed SCOOP model, while other schemes$[i]$ denotes the costs of the competing models under the same experimental conditions. Specifically, if the SCOOP model achieves the lowest cost among all compared schemes in the $i$-th iteration, then $f(i) = 1$; otherwise, $f(i) = 0$. Consequently, the metric $F$ represents the proportion of experimental runs in which the SCOOP model attains the minimum cost. A value of $F$ closer to 1 indicates superior performance of the proposed SCOOP model.

Finally, we compare the performance of the proposed SCOOP model with several representative baseline approaches, including MRDUC [21], MPC-GEMS [22], and HDL [23]. Due to space limitations, only a subset of the experimental results is presented and analyzed in Section 5.2.

## 5.2 Experiment Results

### 5.2.1 *Experiments for Multilinear Tree-based Commitment.*

**Overall Results.** We evaluate the efficiency and scalability of the vector commitment-based DID registration using five

metrics: *Aggregated Proof Generation Time*, *Individual Proof Verification Time*, *Aggregated Proof Verification Time*, *Commitment Update Time*, and *Proof Update Time*. These metrics jointly capture the cost of proof aggregation and verification, as well as the responsiveness of the CPSS to frequent DID updates under different vector sizes. In particular, due to the multilinear tree structure, updating a proof only affects the path from the modified leaf to the root, resulting in an update complexity of $O(w)$. Moreover, the IPA-based proof aggregation procedure incurs a time complexity of $O(wh)$, where $w$ denotes the vector size and $h$ is the tree height.

Fig. 5 illustrates the performance of the multilinear tree-based commitment scheme for vector sizes $w \in \{2, 4, 8, 16, 32, 64, 128, 256, 1024, 2048\}$. The commitment and proof update times are obtained by averaging over 1024 update operations. As shown in Fig. 5(a), with a fixed Merkle tree height $h = 30$, the aggregated proof generation time increases from $0.04$ s at $w = 4$ to $19.85$ s at $w = 2048$, while the individual proof verification time grows from $0.02$ s to $7.66$ s. Fig. 5(b) shows that the commitment update time ranges from $0.02$ s to $4.93$ s, and updating all proofs takes between $0.03$ s and $12.79$ s as $w$ increases. Overall, the proof update, aggregation, and verification costs scale sublinearly with respect to the vector size, indicating that the proposed scheme is well-suited for large-scale DID registries with frequent updates.

**Compared with Merkle SNARK.** To further assess the efficiency of the proposed multilinear tree-based vector commitment, we compare it with the Merkle SNARK–based commitment adopted in representative SSI systems [12]. We observe that when the Merkle tree height reaches $h = 30$, the Merkle SNARK approach requires public parameters of approximately 50 GB, and the parameter generation process takes more than 30 hours. Due to this prohibitive setup cost, we evaluate both schemes under moderate and practical configurations. Specifically, for vector sizes $w \in \{8, 16, 32, 64\}$, we vary the tree height $h \in \{3, 5\}$ for both the Merkle tree and the multilinear tree to balance expressiveness and computational feasibility.

Table 3 summarizes the time performance of the two schemes in terms of three key operations: aggregated proof generation, proof verification, and commitment generation. For aggregated proof generation, the Merkle SNARK scheme exhibits increasing costs as both $h$ and $w$ grow, while the proposed scheme consistently achieves significantly lower runtimes across all configurations. In particular, our approach maintains sub-second aggregation times even at larger vector sizes, demonstrating superior efficiency. For proof verification, the Merkle SNARK scheme provides constant-time verification across different parameters, whereas the proposed scheme incurs slightly higher but still modest verification costs that increase with $h$ and $w$. Despite this increase, verification remains efficient and scales well within practical parameter ranges. In terms of commitment generation, the Merkle SNARK approach incurs substantially higher costs, with runtime growing rapidly as the vector size and tree height increase. By contrast, the proposed multilinear tree–based scheme achieves much lower commitment times, improving scalability by more than an order of magnitude under comparable settings.

Overall, the results indicate that while both schemes support efficient verification, the proposed DID registration scheme significantly outperforms the Merkle SNARK approach in aggregated proof generation and commitment efficiency. Combined with the efficient proof aggregation and update mechanisms shown in Fig. 5 and Table 3, these results demonstrate that SCOOP scales effectively to large CPSS environments with frequent DID updates and verification requests.

#### 5.2.2 Experiments for OSTS.

We present a concise analysis based on experiment settings configured in Section 5.1. Fig. 6 illustrates representative results from the first experimental group. These experiments are conducted under Setting 1, with each sub-configuration (1-1, 1-2, 1-3, 1-4, 1-5, and 1-6) repeated 1000 times. The expected storage cost for each model is obtained by averaging over all iterations.

Fig. 6(a) reports the expected storage cost of the four models under Setting 1-1. The results demonstrate that SCOOP-IC and MRDUC consistently achieve the lowest storage costs. Compared with MPC-GEMS, the proposed OSTS reduces the expected storage cost by an average of 12.4 units. Fig. 6(b) compares the total computation time required to derive the optimal storage strategy under Setting 1-1. For clarity, the computation time is presented on a logarithmic scale. As shown in Fig. 6(b), the OSTS reduces computation time by 95.15% relative to MRDUC. In addition, the average computation time of OSTS is only 1.168 times that of MPC-GEMS. Fig. 6(c) further shows that both OSTS and MRDUC generate optimal task scheduling strategies. Compared with MPC-GEMS, the proposed OSTS achieves an average storage cost reduction of 11.2 units.

Fig. 7 presents representative results from the configuration of Setting 2. For each sub-configuration (2-1, 2-2, 2-3, 2-4, 2-5, and 2-6), the experiments are repeated 1000 times. As shown in Fig. 7(a), both the OSTS and MRDUC consistently generate optimal task scheduling strategies. Compared with MPC-GEMS, the OSTS reduces the expected storage consumption by an average of 12.8 units. Fig. 7(b) illustrates the variation in computation time as the number of tasks increases. The computation time of OSTS decreases with growing task scale, whereas the computation time of MRDUC increases steadily. It indicates that the OSTS better reflects practical deployment conditions in large-scale systems. Fig. 7(c) reports the $F$-index values under selected configurations from Setting 1 and Setting 2. For each configuration, 1000 iterations are performed, and the $F$ index is computed according to Eq. 12. As shown in Fig. 7(c), the $F$ values are generally greater than 0.5, and under Setting 1-2, the $F$ index approaches 1. These results indicate that the proposed OSTS achieves consistent performance advantages over alternative approaches.

Overall, the experimental results demonstrate that the OSTS of SCOOP outperforms the competing models. Although both OSTS and MRDUC achieve comparable optimal storage consumption, OSTS exhibits significantly lower computation overhead as the number of tasks increases, highlighting its superior scalability.

TABLE 3: Performance Comparison between Merkle SNARK and Multilinear Tree-based Commitment Scheme.

| Metric | Scheme | $w=8$ | | | $w=16$ | | | $w=32$ | | | $w=64$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $n$=3 | $n$=4 | $n$=5 | $n$=3 | $n$=4 | $n$=5 | $n$=3 | $n$=4 | $n$=5 | $n$=3 | $n$=4 | $n$=5 |
| Aggregate proofs (s) | Merkle SNARK | 0.32 | 0.38 | 0.44 | 0.63 | 0.76 | 0.86 | 1.27 | 1.49 | 1.73 | 2.54 | 2.98 | 3.44 |
| | Ours | **0.02** | **0.02** | **0.03** | **0.03** | **0.03** | **0.04** | **0.05** | **0.06** | **0.07** | **0.08** | **0.10** | **0.14** |
| Verify (s) | Merkle SNARK | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 |
| | Ours | 0.01 | 0.01 | 0.01 | 0.02 | 0.02 | 0.02 | 0.03 | 0.03 | 0.04 | 0.05 | 0.05 | 0.06 |
| Commitment (s) | Merkle SNARK | 11.88 | 18.32 | 19.30 | 22.99 | 36.39 | 38.02 | 44.30 | 68.59 | 71.67 | 82.90 | 131.32 | 138.17 |
| | Ours | **0.65** | **0.70** | **0.82** | **1.23** | **1.24** | **1.50** | **2.27** | **2.43** | **2.86** | **4.10** | **4.59** | **5.51** |

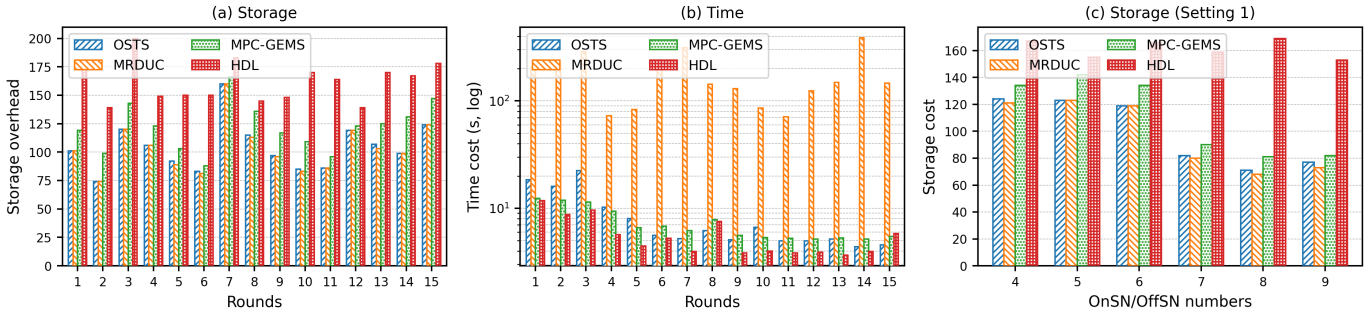


Fig. 6: Comparison results under Setting 1.

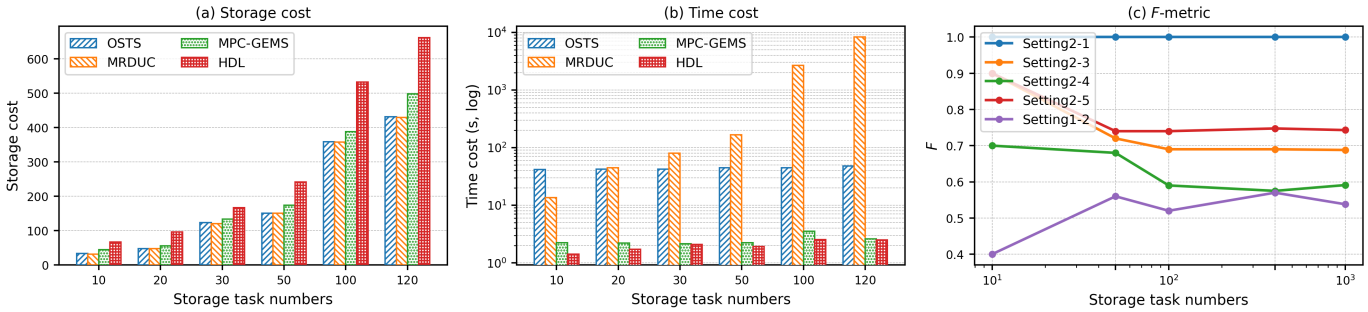


Fig. 7: Performance comparison under Setting 2 with varying storage task numbers. (a) Storage cost. (b) Time cost. (c) $F$ value.

TABLE 4: Time and Storage Overhead of Cryptographic Primitives

| Height ($h$) | Cryptographic Primitives | Time | $vk$ | $pk$ | Proof |
|---|---|---|---|---|---|
| $h=1$ | *MiMC* | 7.937 ms | 2.86 KB | 392 KB | 855 B |
| | *Poseidon* | 8.015 ms | 2.85 KB | 248 KB | 853 B |
| | *Pedersen* | 7.205 ms | 2.86 KB | 3.6 MB | 854 B |
| | *SHA256* | 6.480 ms | 2.85 KB | 31.2 MB | 857 B |
| $h=2$ | *MiMC* | 8.562 ms | 2.86 KB | 1.21 MB | 853 B |
| | *Poseidon* | 7.744 ms | 2.86 KB | 678 KB | 855 B |
| | *Pedersen* | 7.949 ms | 2.86 KB | 11.2 MB | 853 B |
| | *SHA256* | 6.502 ms | 2.86 KB | 93.4 MB | 856 B |
| $h=3$ | *MiMC* | 7.555 ms | 2.86 KB | 2.73 MB | 856 B |
| | *Poseidon* | 7.372 ms | 2.86 KB | 1.50 MB | 856 B |
| | *Pedersen* | 6.492 ms | 2.86 KB | 25.4 MB | 857 B |
| | *SHA256* | 6.652 ms | 2.85 KB | 217.7 MB | 859 B |
| $h=4$ | *MiMC* | 7.571 ms | 2.86 KB | 5.78 MB | 853 B |
| | *Poseidon* | 6.284 ms | 2.85 KB | 3.18 MB | 854 B |
| | *Pedersen* | 6.106 ms | 2.86 KB | 53.8 MB | 852 B |
| | *SHA256* | 6.808 ms | 2.86 KB | 466.7 MB | 857 B |

### 5.2.3 *Experiments for OSS*

Table 4 reports the time cost and storage overhead of different cryptographic primitives in Merkle-based Groth16 constructions. In our evaluation, the verification time is used as the latency constraint for the OSS algorithm, while the sizes of the verification key ($vk$) and proving key ($pk$) are treated as storage overhead. Specifically, the table summarizes the verification time, verification key size ($vk$), proving key size ($pk$), and proof size under different tree heights $h$ for four representative cryptographic primitives, namely MiMC, Poseidon, Pedersen, and SHA256.

The results indicate that the verification time remains relatively stable across different primitives and tree heights, whereas the storage overhead exhibits significant variation. In particular, SHA256 incurs a prohibitively large proving key size as the tree height increases, while MiMC and Poseidon maintain substantially smaller storage footprints, demonstrating their advantage in Merkle-based zk-SNARK constructions. Thus, Table 4 serves as an example of the OSS input table (OST). Given the latency constraint imposed by the verification time, the OSS algorithm derives the optimal storage strategy for different cryptographic primitives by jointly considering the storage overhead of $vk$ and $pk$.

In addition, Fig. 8 presents representative results from

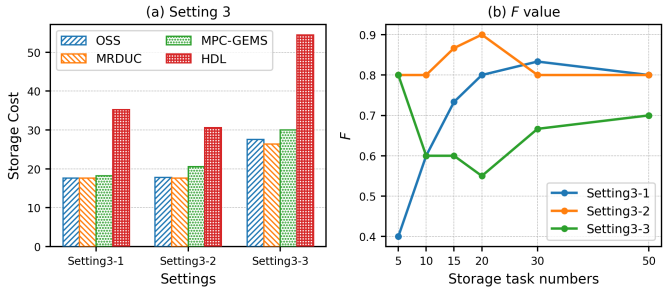


Fig. 8: Experiment Results under Setting 3. (a) Storage cost comparison. (b) $F$ value under varying storage task numbers.

the third group of experiments. For each sub-setting in Setting 3, we conduct 1000 independent iterations and report the averaged results. As shown in Fig. 8(a), both the proposed OSS and MRDUC consistently derive optimal storage strategies. Compared with MPC-GEMS, OSS reduces the expected storage resource cost by an average of 9.2%, demonstrating its effectiveness in minimizing storage overhead under the same constraints. To further evaluate the performance, we compute the index $F$ according to Eq. 12 based on 1000 iterations for each configuration. Fig. 8(b) shows that the value of $F$ is consistently greater than 0.5 across different sub-settings, indicating a stable performance advantage of SCOOP over alternative schemes.

Consequently, the experiment results confirm that the proposed OSS outperforms competing approaches. By optimally scheduling storage tasks between OnSNs and OffSNs, SCOOP effectively reduces the storage cost of computing devices. Moreover, OSS is able to generate optimal storage strategies tailored to different cryptographic primitives, providing flexible and efficient support for DID Holders in heterogeneous deployment environments.

## 6 Related Work

**Decentralized Identity.** DID is an identity authentication and management paradigm built upon distributed systems and cryptographic techniques [24], [25]. Unlike traditional centralized identity management architectures, DID eliminates reliance on a single trusted authority by distributing identity-related data across multiple nodes [26]. In such systems, users retain full control over their identity information and leverage cryptographic mechanisms to ensure data integrity, confidentiality, and privacy.

Recent studies have explored the application of DID in diverse network environments. Xiong et al. [27] proposed a blockchain-based decentralized identity management framework for Vehicular Ad-hoc Networks (VANETs) and large-scale Internet of Things (IoT) systems. Kara et al. [28] introduced a decentralized identity authentication mechanism for Voice over Internet Protocol (VoIP) networks, aiming to mitigate single points of failure and privacy leakage issues inherent in centralized authentication schemes. Similarly, [29] presented a user-centric authentication protocol for electric vehicle charging scenarios based on blockchain-enabled decentralized identifiers. Existing approaches primarily focus on deploying DID mechanisms across different networking contexts, addressing challenges such as single points of failure and privacy protection in identity management [30]. In contrast to prior studies emphasizing adaptive blockchain storage strategies [31], [32] and privacy-aware DID architectures [33], [15], the present work concentrates on security issues arising during identity registration and update processes in DID systems, which remain relatively underexplored.

**On-chain and Off-chain Data Storage.** On-chain and off-chain data storage is a widely adopted data management strategy in DID systems [34]. In this paradigm, critical metadata or cryptographic hashes are stored on the blockchain, while the actual identity data are maintained off-chain, thereby reducing blockchain storage overhead while preserving data integrity and verifiability [7], [35], [36]. Previous studies have explored various governance strategies for coordinating on-chain and off-chain storage to achieve scalable and efficient data management in decentralized systems [8]. For instance, [9] demonstrated that such a hybrid storage approach enables fine-grained transaction control through a multi-layer storage architecture. Similarly, [37] combined Trusted Execution Environments (TEEs) with on-chain and off-chain mechanisms to support trustworthy data collection and processing. In addition, several studies have investigated dynamic resource scheduling algorithms in blockchain–IoT systems [38], [39]. However, in contrast to SCOOP, these approaches typically consider only a single-stage task allocation and do not account for the impact of heterogeneous cryptographic primitives on storage consumption in blockchain-enabled environments. In line with these efforts, our work adopts an on-chain/off-chain storage strategy as a fundamental building block for constructing a scalable DID implementation, while focusing on security and efficiency considerations in identity lifecycle management.

Verifiable authentication mechanisms have been extensively investigated in blockchain-enabled systems to enhance trust, privacy, and accountability. Sallal et al. [40] employed the Selene voting scheme together with permissioned blockchains to achieve election verifiability. However, this approach relies on complex cryptographic primitives and intricate system design, leading to high implementation complexity. Moreover, voter verification depends entirely on the secure management of tracking codes; loss or compromise of these codes may prevent successful vote verification. Deebak et al. [41] proposed a privacy-preserving seamless authentication scheme based on provable key generation. Fotiou et al. [42] introduced a capability-based access control framework for IoT devices leveraging verifiable credentials. Mazzocca et al. [43] designed an efficient mechanism for verifiable credential invocation in IoT networks, while Patel et al. [44] proposed a user-empowered, privacy-preserving authentication scheme tailored for digital twin environments.

Despite their effectiveness, many of these schemes rely on cryptographic constructions such as bilinear aggregate signatures and verifiable secret sharing, which incur substantial computational overhead. This limitation becomes particularly pronounced in large-scale and resource-constrained distributed systems. Furthermore, bilinear pairing-based schemes typically involve long public

and private keys, increasing the costs of key storage and transmission. In verifiable secret sharing protocols, participants may also deny their actions or secret shares, resulting in weakened non-repudiation guarantees.

## 7 Conclusions

Based on blockchain technology, DID is emerging as the future direction for IoT and Web3.0. A key challenge in providing informed decision-making through IoT is to offer efficient and energy-saving solutions that cater to both on-chain and off-chain storage for DIDs. This paper has sparked a demand for efficient task publishing and storage strategy generation, developing a model for scalable on-chain and off-chain storage processes for DIDs in CPSS. The SCOOP model comprises the application layer, decision layer, and storage layer. IC, the core component of the application layer, handles data collection, filtering, and forwarding. To deliver high-performance storage services, the decision layer's IC and OnSN/OffSN execute the OSTS algorithm and OSS algorithm, respectively. The OSTS algorithm optimizes energy consumption by assigning tasks to suitable OnSN/OffSN under time constraints. The experiments have verified the efficiency and rationality of our approach in task allocation and DID storage decision-making.

Although the SCOOP scheme provides an efficient on-chain–off-chain storage optimization framework for DID, its current design is primarily oriented toward a single blockchain. In real-world deployment, DIDs often need to operate across multiple blockchains or administrative domains. Different domains may implement distinct access control rules, storage policies, or cryptographic standards, which may make it difficult to consistently apply the optimal storage strategy generated by OSTS and OSS in cross-chain scenarios. Additionally, cross-chain verification delays can introduce non-negligible time overhead, necessitating additional constraints. Future work should extend SCOOP to support cross-chain interoperability protocols and policy-aware adaptive scheduling to coordinate storage strategies across technological boundaries.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No.s U24B20146, 62372044), Beijing Municipal Science and Technology Commission Project (Z241100009124008), and Beijing Nova Program (Grant No. 20250484921).

## References

[1] O. Avellaneda, A. Bachmann, A. Barbir, J. Brenan, P. Dingle, K. H. Duffy, E. Maler, D. Reed, and M. Sporny, "Decentralized identity: Where did it come from and where is it going?" *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 10–13, 2019.

[2] O. Dib and B. Rababah, "Decentralized identity systems: Architecture, challenges, solutions and future directions," *Annals of Emerging Technologies in Computing*, vol. 4, no. 5, pp. 19–40, 2020.

[3] S. Pasandideh, P. Pereira, and L. Gomes, "Cyber-physical-social systems: taxonomy, challenges, and opportunities," *IEEE Access*, vol. 10, pp. 42 404–42 419, 2022.

[4] X. Zhou, W. Liang, J. Ma, Z. Yan, and K. I.-K. Wang, "2d federated learning for personalized human activity recognition in cyber-physical-social systems," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 6, pp. 3934–3944, 2022.

[5] M. Sookhak, M. R. Jabbarpour, N. S. Safa, and F. R. Yu, "Blockchain and smart contract for access control in healthcare: A survey, issues and challenges, and open issues," *Journal of Network and Computer Applications*, vol. 178, p. 102950, 2021.

[6] B. Zaabar, O. Cheikhrouhou, F. Jamil, M. Ammi, and M. Abid, "Healthblock: A secure blockchain-based healthcare data management system," *Computer Networks*, vol. 200, p. 108500, 2021.

[7] J. Hao, C. Huang, W. Tang, Y. Zhang, and S. Yuan, "Smart contract-based access control through off-chain signature and on-chain evaluation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 4, pp. 2221–2225, 2021.

[8] C. Xu, C. Zhang, J. Xu, and J. Pei, "Slimchain: Scaling blockchain transactions through off-chain storage and parallel processing," *Proceedings of the VLDB Endowment*, vol. 14, no. 11, pp. 2314–2326, 2021.

[9] T. Cai, W. Chen, K. E. Psannis, S. K. Goudos, Y. Yu, Z. Zheng, and S. Wan, "Scalable on-chain and off-chain blockchain for sharing economy in large-scale wireless networks," *IEEE Wireless Communications*, vol. 29, no. 3, pp. 32–38, 2022.

[10] N. Sangeeta and S. Y. Nam, "Blockchain and interplanetary file system (ipfs)-based data storage system for vehicular networks with keyword search capability," *Electronics*, vol. 12, no. 7, p. 1545, 2023.

[11] M. Kaur, S. Gupta, D. Kumar, M. S. Raboaca, S. Goyal, and C. Verma, "Ipfs: An off-chain storage solution for blockchain," in *Proceedings of International Conference on Recent Innovations in Computing: ICRIC 2022, Volume 1*. Springer, 2023, pp. 513–525.

[12] R. Mukta, H.-Y. Paik, Q. Lu, and S. S. Kanhere, "Credtrust: Credential based issuer management for trust in self-sovereign identity," in *2022 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2022, pp. 334–339.

[13] H. Xie, D. Ding, L. Zhao, K. Kang, and Q. Liu, "A two-stage preference driven multi-objective evolutionary algorithm for workflow scheduling in the cloud," *Expert Systems with Applications*, vol. 238, p. 122009, 2024.

[14] X. Zhou, W. Liang, I. Kevin, K. Wang, and S. Shimizu, "Multi-modality behavioral influence analysis for personalized recommendations in health social media environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 888–897, 2019.

[15] B. Chen, X. Liu, H. Xu, S. Chen, and K. Li, "Bssn: Enabling adjustable blockchain storage for resource-constrained iot scenarios," *IEEE Internet of Things Journal*, 2024.

[16] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Computing*, vol. 24, no. 1, pp. 205–223, 2021.

[17] K. Gai, Q. Xiao, M. Qiu, G. Zhang, J. Chen, Y. Wei, and Y. Zhang, "Digital twin-enabled ai enhancement in smart critical infrastructures for 5g," *ACM Transactions on Sensor Networks (TOSN)*, vol. 18, no. 3, pp. 1–20, 2022.

[18] K. Gai, Y. Zhang, M. Qiu, and B. Thuraisingham, "Blockchain-enabled service optimizations in supply chain digital twin," *IEEE Transactions on Services Computing*, vol. 16, no. 3, pp. 1673–1685, 2022.

[19] S. Srinivasan, A. Chepurnoy, C. Papamanthou, A. Tomescu, and Y. Zhang, "Hyperproofs: Aggregating and maintaining proofs in vector commitments," in *31st USENIX Security Symposium, USENIX Security 2022*, Boston, MA, USA, 2022, pp. 3001–3018.

[20] L. F. Zhang and H. Wang, "Multi-server verifiable computation of low-degree polynomials," in *43rd IEEE Symposium on Security and Privacy, SP 2022*, San Francisco, CA, USA, 2022, pp. 596–613.

[21] H. Xiong, Y. Shi, Z. Chen, C. Guo, and Y. Ding, "Multi-stage robust dynamic unit commitment based on pre-extended-fast robust dual dynamic programming," *IEEE Transactions on Power Systems*, vol. 38, no. 3, pp. 2411–2422, 2022.

[22] Y. Xie, Y. Ueda, and M. Sugiyama, "Greedy energy management strategy and sizing method for a stand-alone microgrid with hydrogen storage," *Journal of Energy Storage*, vol. 44, p. 103406, 2021.

[23] R. Rathi, N. Narkhede, and M. Hasamnis, "Design and implementation of first-in-first-out (fifo) buffer for distributed load balancing systems," in *International Conference on Smart Computing and Communication*. Springer, 2024, pp. 11–21.

[24] E. Samir, H. Wu, M. Azab, C. Xin, and Q. Zhang, "Dt-ssim: A decentralized trustworthy self-sovereign identity management

framework," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 7972–7988, 2022.
[25] T. Xie, K. Gai, L. Zhu, Y. Guo, and K.-K. R. Choo, "Cross-chain-based trustworthy node identity governance in internet of things," *IEEE Internet of Things J.*, vol. 10, no. 24, pp. 21 580–21 594, 2023.
[26] T. Xie, K. Gai, L. Zhu, S. Wang, and Z. Zhang, "Rac-chain: An asynchronous consensus-based cross-chain approach to scalable blockchain for metaverse," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 20, no. 7, pp. 1–24, 2024.
[27] R. Xiong, W. Ren, X. Hao, J. He, and K.-K. R. Choo, "Bdim: A blockchain-based decentralized identity management scheme for large scale internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 22 581–22 590, 2023.
[28] M. Kara, H. R. Merzeh, M. A. Aydın, and H. H. Balık, "VoIPChain: A decentralized identity authentication in voice over IP using blockchain," *Computer Communications*, vol. 198, pp. 247–261, 2023.
[29] R. P. Parameswarath, P. Gope, and B. Sikdar, "Privacy-preserving user-centric authentication protocol for iot-enabled vehicular charging system using decentralized identity," *IEEE Internet of Things Magazine*, vol. 6, no. 1, pp. 70–75, 2023.
[30] B. Chen, S. Waiwitlikhit, I. Stoica, and D. Kang, "ZKML: an optimizing system for ML inference in zero-knowledge proofs," in *Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024*, Athens, Greece, 2024, pp. 560–574.
[31] J. Xu, Q. Xie, S. Peng, C. Wang, and X. Jia, "Adaptchain: Adaptive scaling blockchain with transaction deduplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 6, pp. 1909–1922, 2023.
[32] X. Zhou, W. Huang, W. Liang, Z. Yan, J. Ma, Y. Pan, and K. I.-K. Wang, "Federated distillation and blockchain empowered secure knowledge sharing for internet of medical things," *Information Sciences*, vol. 662, p. 120217, 2024.
[33] E. Zeydan, L. Blanco, J. Mangues-Bafalluy, S. S. Arslan, Y. Turk, A. K. Yadav, and M. Liyanage, "Blockchain-based self-sovereign identity: Taking control of identity in federated learning," *IEEE Open Journal of the Communications Society*, 2024.
[34] K. Ding, T. Xie, K. Gai, C. Guo, L. Lei, D. Wang, J. Yu, L. Zhu, and W. Meng, "Verifiable decentralized identity-based meta-computing in industrial internet of things (iiot)," *Journal of Systems Architecture*, vol. 162, p. 103391, 2025.
[35] Z. Fang, J. Yu, G. Huang, R. Dong, and K. Gai, "Promise: A pedersen commitment-based transaction hiding scheme for blockchain system," in *Blockchain and Web3 Technology Innovation and Application Exchange Conference*. Springer, 2025, pp. 177–188.
[36] X. Zhou, J. Wu, W. Liang, K. I.-K. Wang, Z. Yan, L. T. Yang, and Q. Jin, "Reconstructed graph neural network with knowledge distillation for lightweight anomaly detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 9, pp. 11 817–11 828, 2024.
[37] C. Liu, H. Guo, M. Xu, S. Wang, D. Yu, J. Yu, and X. Cheng, "Extending on-chain trust to off-chain–trustworthy blockchain data collection using trusted execution environment (tee)," *IEEE Transactions on Computers*, vol. 71, no. 12, pp. 3268–3280, 2022.
[38] Y. Zhang, Y. Liang, B. Jia, and P. Wang, "Scheduling and process optimization for blockchain-enabled cloud manufacturing using dynamic selection evolutionary algorithm," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1903–1911, 2022.
[39] K. Lin, J. Gao, G. Han, H. Wang, and C. Li, "Intelligent blockchain-enabled adaptive collaborative resource scheduling in large-scale industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 9196–9205, 2022.
[40] M. Sallal, R. de Fréin, and A. Malik, "Pvpbc: Privacy and verifiability preserving e-voting based on permissioned blockchain," *Future Internet*, vol. 15, no. 4, p. 121, 2023.
[41] B. D. Deebak and S. O. Hwang, "Privacy preserving based on seamless authentication with provable key verification using miomt for b5g-enabled healthcare systems," *IEEE Transactions on Services Computing*, vol. 17, no. 3, pp. 1097–1113, 2024.
[42] N. Fotiou, V. A. Siris, G. C. Polyzos, Y. Kortesniemi, and D. Lagutin, "Capabilities-based access control for iot devices using verifiable credentials," in *43rd IEEE Security and Privacy, SP Workshops 2022*, San Francisco, CA, USA, 2022, pp. 222–228.
[43] C. Mazzocca, A. Acar, A. S. Uluagac, and R. Montanari, "EVOKE: efficient revocation of verifiable credentials in iot networks," in *33rd USENIX Security Symposium, USENIX Security 2024*, Philadelphia, PA, USA, 2024, p. 99.
[44] C. Patel, A. M. Pasikhani, P. Gope, and J. A. Clark, "User-empowered secure privacy-preserving authentication scheme for digital twin," *Computers & Security*, vol. 140, no. 99, p. 103793, 2024.