Verifiable Decentralized Identity-based Meta-computing in Industrial Internet of Things (IIoT)

Kai Ding[a], Tianxiu Xie[a], Keke Gai[a], Chennan Guo[a], Liangqi Lei[a], Dongjue Wang[a], Jing Yu[b], Liehuang Zhu[a] and Weizhi Meng[c]

[a]School of Cyberspace Science and Technology, Beijing Institute of Technology, China,
[b]School of Information Engineering, Minzu University of China, China
[c]School of Computing and Communications, Lancaster University, United Kingdom

ARTICLE INFO

Keywords:
Decentralized identity
Verifiable registry
Meta-computing
Trustworthy authentication
Blockchain

ABSTRACT

Meta-computing in Industrial Internet of Things (IIoT) has triggered a dramatic advance due to the gigantic supports of computation power for processing complex IIoT tasks. However, users identities are encountering security and verification issues since emerging threats derive from dynamic inter-operations in the cross-organization context. Even though blockchain-based Decentralized Identity (DID) is an alternative for offering a strengthened identity governance, current verifiability of DID documents still encounters vulnerabilities due to the involvement of the less trustful third parties that maintain the storage of binding relationships between DID identifiers and public keys. In this paper, we propose a novel *Verifiable and Searchable Decentralized Identity (VS-DID)* model. We focus on the verifiability of DID documents and propose a verifiable registry scheme that ensures verifiable binding relationships. In order to enable efficient queries in large-scale users' identities in meta-computing IIoT, we develop an on-chain-off-chain query strategy that adopts a slide window accumulator. The experimental results show that our scheme reduces aggregate proof time and commitment time by 93.5% and 96.5%, respectively, compared to the Merkle SNARK scheme, while maintaining reasonable verification time, significantly improving the efficiency of DID registry in large-scale IIoT environments.

1. Introduction

The emergency of meta-computing has been powering up computing and service models of contemporary *Industrial Internet of Things* (IIoT) due to the implementation of multiple technical characteristics, e.g., blockchain native and open access. It enables distributed computing power via utilizing the increasing computation and storage capabilities of *Internet of Things* (IoT) devices to achieve a higher-level computing power integration [47, 54, 20]. Considering a wide access setting in meta-computing-empowered IIoT, one of the challenges is to secure the verification of users' identities within the multi-system context [21, 25].

An alternative solution is to explore the implementation of *Decentralized Identity* (DID), which leverages advanced data storage diagram techniques to enhance the efficiency and reliability of data storage [43, 30]. The existing work in DID has implied a great transformative potential in revolutionizing identity verification practices across diverse sectors, e.g., in finance, healthcare, and logistics [5]. To be specific, hashed data are stored on-chain in a Merkle tree while a complete copy of data is off-chain stored. This paradigm reduces the amount of data stored in blocks as only hash values are stored on-chain. Thus, the maintenance of on-chain data storage is lowered down, which facilitates a large-scale off-chain data storage and guarantees data integrity and tamper-resistant.

However, existing schemes lack verifying DID documents and rise the cost of data maintenance, which causes trouble in meeting the demands of querying large-scale data. In W3C standards [44], for example, the binding of DID identifiers and public keys are stored in the DID document after the user registry. A potential risk exist as DID documents are maintained by a third party but the binding relations need verification, which causes a conflict when covering privacy concerns [15, 32]. As cross-domain/organization user authentication is a common requirement in meta-computing, unexpected releases of relations threaten the security of users' identities. Moreover, computation and maintenance maybe costly, since updating on-chain data requires re-computing tree roots in the current Merkle tree-based storage model. The challenge becomes further greater when encountering large data workloads in the IIoT context. These issues have been rarely addressed by previous research.

In order to address challenges above, in this paper we focus on the issue of verifiable registry of decentralized identity and propose a novel *Verifiable and Searchable Decentralized Identity* (VS-DID) model in order to realize the verifiability of DID documents and support efficient queries in large-scale user identity systems. Fig. 1 illustrates a high-level architecture of the proposed VS-DID model. In essence, in our proposed approach, blockchain-based DID utilizes a network of nodes to store identity information, allowing each individual/object to control and manage identity data. We highlight the effect of all participants' accesses (either physically or virtually) in meta-computing-enabled IIoT. Specifically, the purpose of employing DID in meta-computing IIoT is to guarantee the security of identity governance while ensuring a proper adaptability, e.g., high efficiency and low computing costs. Cross-domain/organization

*T. Xie is the corresponding author.

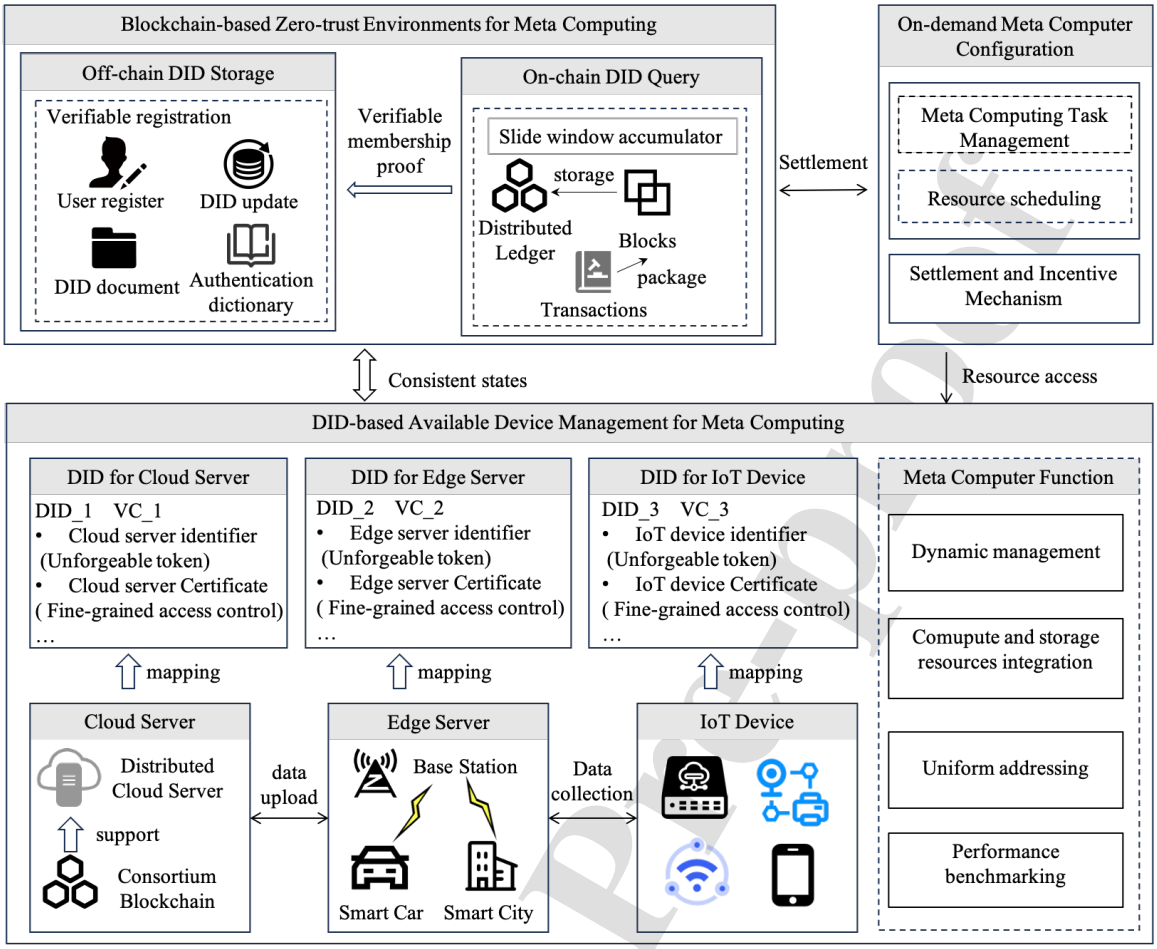

Figure 1: The high-level architecture of the proposed VS-DID model for meta-computing-enabled IIoT.

requirements are covered by the proposed model, which covers both users and hardware/software.

In our approach, both on-chain and off-chain storage are applied in order to reduce the workload of on-chain storage, which is one of the key elements influencing the performance of meta-computing in IIoT. Considering the verifiability of the third-party owning DID documents, our approach develops a verifiable registry to ensure the binding relations between the user key pairs and DID identifiers. The module mainly consists of three parts, namely, *User Register*, *DID Update*, and *DID Document*. The binding relations between DID identifiers and public keys are verified through membership verification proof. The verifiable registry is based on a bilinear tree-based commitment to maintain in-variance proof. DID document is an extension of the *Key-Value Commitment* (KVaC) dictionary structure [3] that binds key-value pairs to a commitment, allowing proof of their existence and consistency without revealing the specific key-value content.

To address the high maintenance cost issue discussed above, we propose a slide-window accumulator-based storage mechanism, in which uses a hierarchical storage setting so that users only store DID and *Verifiable Credentials* (VCs) on the blockchain and leave other data stored off-chain. That is to say, the proposed accumulator improves the performance of DID verifier during the query verification process, whenever a user or DID verifier needs to perform a query/verification. An algorithm is developed for maintaining the sliding window accumulator and handling user queries. The design goals of the proposed approach cover efficiency of identity query, storage saving, and identity in-time updates.

The main contribution are as summarized as follows.

- We have proposed a novel verifiable registry scheme that is specifically tailored for DID documents. Our verifiable registry is implemented through key-value commitment technology, so that the integrity and immutability of the identity contents are ensured. The proposed scheme considers the implementation of a wide access in IIoT, which is capable of governing meta-computing identity and authentication within the multi-system context.
- We have developed a sliding window accumulator for achieving efficient data queries for both on-chain and off-chain, covering boolean queries and range queries.

The proposed accumulator can successfully support not only DID implementations but also complex application requirements in IIoT, i.e., meta-computing-enabled resource integration and large volume data processing cross distinct systems.

- We have conducted a theoretical analysis in security and efficiency and implemented experiment evaluations for evaluating the performance of the proposed accumulator. Our evaluation results have demonstrated that our approach has an advantage in offering secure verifiability and high efficiency, comparing to other existing methods.

The remainder of this paper is organized as follows. Section 3 shows the work's background and Section 4 presents detailed descriptions about the proposed model, covering operating principles and major modules. We provide experiment evaluations and main findings in Section 5. Related work has been briefly reviewed in Section 2. Finally, conclusions are drawn in Section 6.

2. Related Work

Xiong *et al.* [50] proposed a blockchain-based decentralized identity management scheme applied to VANET (Vehicular Ad-hoc Network) and other large-scale IoT networks. Kara *et al.* [24] introduced a blockchain-based decentralized network identity authentication mechanism for VoIP (Voice over Internet Protocol), which addressed identity verification failure issues in network communication due to single points of failure and privacy concerns. This mechanism outperforms the TLS [27] and SIP [23] protocols in terms of performance and security. Parameswarat *et al.* [35] proposed a user-centric electric vehicle charging authentication protocol that utilizes blockchain-based decentralized identifiers. This approach combines VC with DID to provide zero-knowledge proofs for users [11], enabling users to maintain full control over their identities, thus facilitating privacy-preserving user-centric authentication.

We observed that prior methods mainly focused on exploring the mechanism of DID in various networking environments, addressing concerns such as single points of failure and privacy protection in identity management [11]. Our prior work also explored DID applications and proposed a cross-chain scheme to ensure the legitimacy of IoT node identities [48]. In contrast to the prior work, our current study focuses on addressing security issues in identity registry and updates within DID systems.

On-chain and Off-chain Data Storage. On-chain and off-chain data storage is a data management strategy used in DID systems, where on-chain storage maintains key information or data hashes on the blockchain, while off-chain storage is responsible for actual data maintenance [51, 22, 19]. Previous studies have explored various approaches to governing on-chain and off-chain storage to achieve scalable data management [51]. For example, Cai *et al.* [9] demonstrated that this transferable approach could achieve a fine-grained transaction framework with multi-layer storage designs. Liu *et al.* [29] combined Trusted Execution Environments (TEE) with an on-chain and off-chain strategy to enable trustworthy data collection. Our work aims to leverage the on-chain-off-chain strategy for constructing a scalable DID implementation in meta-computing environments.

Sallal *et al.* [37] employed the Selene voting scheme [4] and permissioned chains in blockchain voting systems to ensure election verifiability. The system generates a unique tracking code for each vote and provides it to the voter. At this stage, the voting content is associated with the tracking code, but there is no record of the voter's identity tied to the tracking code. Voters can verify their votes without revealing their identity, thus ensuring transparency and fairness in the election results. However, this scheme requires complex cryptographic techniques and system design, making implementation challenging. Voter verification depends on the tracking code, so voters must safeguard their tracking codes properly; otherwise, they may not be able to verify their votes. Deebak *et al.* [14] proposed a privacy-preserving seamless authentication mechanism using provable key generation. Fotiou *et al.* [17] introduced a novel capabilities-based access control model for IoT devices using verifiable credentials. Shen *et al.* [39] proposed a privacy-preserving federated learning scheme based on the BCP cryptosystem, which can verify user identity and data integrity in a multi-key environment. This scheme uses bilinear aggregate signatures and verifiable secret sharing to verify user data integrity and identity, effectively excluding erroneous data from some users. Xie *et al.* [53] proposed an anti-disguise authentication system based on the first impression of avatars in the metaverse.

However, bilinear aggregate signatures and verifiable secret sharing techniques have high computational complexity, particularly in large-scale, resource-constrained distributed systems. The public and private keys in bilinear mapping are relatively long, increasing the overhead for key storage and transmission. Additionally, participants in verifiable secret sharing may deny their actions or secret shares, which weakens non-repudiation guarantees. Our scheme utilizes a sliding window accumulator to achieve verifiable identity authentication. This approach reduces computational overhead through incremental updates while ensuring security, thus improving the efficiency of identity authentication.

Various types of cryptographic accumulators have been proposed in the literature. Research has shown that accumulators based on *Elliptic Curve Cryptography* (ECC) [1] outperform RSA-based accumulators and Merkle Trees (MTs). Wang *et al.* [46] proposed cross-domain dynamic accumulators through blockchain in IoT systems. Foteini *et al.* [6] introduced the concept of oblivious accumulators, which ensure privacy by concealing both the accumulated elements and the set size from all parties involved. They presented a construction based on key-value commitments and established lower bounds on the communication required for these and almost-oblivious accumulators. Xin *et*

al. [49] proposed dynamic proofs of liabilities from zero-knowledge RSA accumulators. Liu *et al.* [31] introduced a fine-grained authentication approach for range queries in hybrid-storage blockchains, addressing the limitations of existing coarse-grained solutions. Si *et al.* [42] presented compressed zero-knowledge proofs for lattice-based accumulators. ECC-based accumulators offer several benefits, including smaller membership proofs (witnesses) and more efficient verification due to fewer required mathematical operations.

3. Background

3.1. Meta-computing-enabled IIoT

Meta-computing is an emerging technical concept that aims to integrates multiple technologies to serve those application scenarios requiring trustworthy computing resource sharing, e.g., industrial Internet, metaverse, or Web 3.0, since the operating principle of this technology supports a combination of various technical merits, such as zero trust, computing power sharing, and flexible computing services [12, 40, 18]. Prior studies have pointed out that both hardware and software are objectives that are shared by implementing a role of "manager" within a middle-ware layer. The manager is responsible for a group of tasks, such as estimating and updating states of computing resources, allocating tasks, and offering trustworthy environment [45, 28]. In the IIoT context, a meta-computing-enabled IIoT primarily aims at optimizing the utilization of available computing infrastructure, from cloud/edge servers to IoT devices, while considering trustworthiness and security.

Align with the structure illustrated in Fig. 1, we emphasize the significance of the identity in meta-computing-enabled IIoT, on the basis of our understanding on meta-computing, as the scope of identities cover not only user identities but also devices' identities. The implementation of meta-computing requires inter-connectivities and inter-activities between various servers and devices, as well as multiple groups of participants; therefore, cross-domain access is one of characteristics of meta-computing, i.e., establishing a trustworthy mechanism for identity verifications is a fundamental requirement for achieving functionality of meta-computing. To be specific, in order to realize a meta-computing-enabled IIoT, identity verification module is directly associated with a few representative modules, such as the resource scheduler, trustworthiness manager, access control manager, and incentive controller. In our scheme, we involve identities of users and devices/servers in the scope of verification and develop a DID scheme in order to meet the demand of cross-domain/organization.

Existing identity management solutions in IIoT, such as centralized Public Key Infrastructure (PKI) and federated identity systems [13, 2], face challenges in scalability and trustworthiness due to their reliance on centralized authorities. Recent decentralized approaches [7, 26], including blockchain-based identity systems, offer improved security and transparency but often suffer from high computational overhead and limited interoperability. Our proposed DID scheme addresses these limitations by providing a scalable, interoperable, and trustless identity verification mechanism tailored for meta-computing-enabled IIoT environments.

3.2. Verifiable Decentralized Identity (DID)

Decentralized Identity. DIDs fundamentally transform the decentralized identity recognition framework. A DID uniquely identifies the subject, whether a human or non-human entity, and comprises three elements, namely, a Uniform Resource Identifier (URI) , a specific DID method identifier, and a method-specific DID identifier. Furthermore, a DID URL extends the basic DID by incorporating additional URI components. Each DID resolves to a machine-readable JSON-LD document known as a DID document, which includes cryptographic public keys, service endpoints, authentication parameters, timestamps, and metadata. By eliminating the need for identity providers and centralized authority, DIDs allow entities to prove ownership with a private key corresponding to the public key in the DID document. Verification occurs through accessing the public DID document, shared via a verifiable data registry typically implemented using Distributed Ledger Technology (DLT) [36]. Verifiable credentials [38] is another W3C specification that offers an interoperable data structure for representing encrypted, verifiable, and tamper-evident claims.

Verifiable Authentication. There has been extensive research [10] on verifiable authentication. Dushku *et al.* [16] propose a protocol that does not use public key encryption for IoT publish/subscribe communication paradigms. This protocol uses a one-way key chain allowing multiple verifiers to prove one or more provers without pre-shared key materials. The one-way key chain generates a series of keys through a one-way function, enhancing system security and simplifying key management. However, the length of the key chain is limited. Once all keys in the key chain are used up, a new key chain needs to be regenerated, increasing management complexity. Moreover, in certain application scenarios, users need to stay synchronized with the system. If users and verifiers of the key chain are not synchronized, verification failures may occur.

Moreover, current DID-based authentication schemes, such as EVOKE [34] and CanDID [33], provide robust identity management but face challenges in security and computational efficiency. For instance, CanDID reliance on the zero-knowledge proof techniques introduces latency, while EVOKE's RSA accumulator compromises data security. Our proposed scheme addresses these issues by leveraging a hybrid on-chain-off-chain storage strategy and a sliding window accumulator, significantly improving scalability and verification efficiency.

Cryptographic Accumulators. The cryptographic accumulators [8] are proposed as a way to eliminate reliance on trusted central authorities. Accumulators are used to cryptographically create a short, binding commitment to a set of elements, known as the accumulator value, enabling

the proof of membership within a dataset through compact proofs.

An accumulator is defined as a family of one-way hash functions that exhibit a quasi-commutative property. A one-way hash function h transforms an input value v of arbitrary length into a fixed-size output called the hash value $h(v)$. The one-way nature of this function implies that, given $h(v)$, it is computationally infeasible to recover the original input. The quasi-commutative property allows for the accumulation of values in any order, making accumulators effective for compactly representing a set of elements and for efficiently verifying membership. To determine whether an element v_i is included in the accumulator value, one must compute its corresponding witness w_i, derived by accumulating all values except v_i. Accumulators that verify the inclusion of elements using membership witnesses are called positive accumulators, while those that verify non-inclusion through non-membership witnesses are known as negative accumulators. Accumulators that support both functionalities are referred to as universal accumulators.

4. Proposed Model

4.1. Design goals

We aim to achieve the following design goals in VS-DID.

Efficient Identity Query. We consider the requirement of the frequent and real-time identity verification in meta-computing IIoT, such that an efficient query function for rapid system response is needed. User waiting time shall be shortened for satisfying the requirement of user experience and satisfactions in multiple implementations in IIoT, e.g., financial transactions and remote interactions. The functionality of identity query shall support multi-domain applications within tons of participant IoT devices.

Scalable Secure Storage. We consider the increasing number of users with identity data a potential burden for system storage, along with the growth of meta-computing-enabled IIoT. Due to the distributed setting for both user identities and meta-computing applications, blockchain-based solutions require supports from an effective on-chain-off-chain mechanism in order to meet the demand of large workloads. A scalable storage is needed to reach a balance between data security and storage saving.

Verifiable Timely Update. A meta-computing-enabled IIoT generally consists of multiple subsystems for dealing with distinct tasks, which leads to dynamic updates for users' identities. The attributes used for defining VCs maybe frequently updated within a certain system context, so that a secure verification for identity updates is needed. A conflict often exist in weighting security and efficiency. Thus, we aim to achieve a timely update on identity information while preventing verification errors and other technical risks caused by outdated information.

4.2. Model Design

In our VS-DID model, identity registry is designed for matching the requirements of meta-computing in IIoT, in which supports trustworthy verifications in the cross-domain context. First, users register their identities through a DID system by which a DID Verifier verifies the user's identity from a registry application. A DID can be issued by a DID Issuer when the registry request is successfully verified. We assume that a VC is issued by a trusted party (a DID Issuer), so that a VC is used to prove the authenticity of the user's identity in subsequent operations. For trustworthy verification, a verifiable registry is maintained by the DID Issuer, which stores encrypted key-value pairs (each registered user's DID and public key) and processes digital signatures.

The verifiable registry applies a key-value commitment-based authenticated dictionary, in which the key-value pair of the user's public key and DID are bound to a commitment. This commitment can be made public for verifiers to check the existence and consistency of the data. When a new user registers, the new key-value pair will be paired with a commitment to ensure its immutability and integrity. When a user's public key changes, the system will calculate a new DID value and commitment. The user's public key and DID are paired with the commitment; then, the key-value pair in the registry is updated.

In addition, the verifier uses a public key provided by the user as the key and the user's DID as the value, when the DID Verifier checks the user's key-value pair in the verifiable registry. The verifier then measures the key-value pair with the previously published commitment. When the provided key-value pair matches the content in the commitment, it will prove that the key-value exists in the the original data and is bound to the commitment at a certain time. Therefore, the DID Verifier authenticates the user's identity.

In this work, we propose a hybrid on-chain and off-chain storage approach to ensure information integrity verification while minimizing blockchain storage usage and overall system consumption, All users' DIDs and VCs information are stored off-chain, while only the computed root hash values are stored on-chain.

A typical VC contains a large number of keywords related to user behavior and capabilities. This work extracts the keywords from VC and aggregates them through data aggregation, forming a data object with the aggregated keyword set. Storing all data objects on the blockchain would consume significant resources and result in slow query speeds for traversing all data objects. Therefore, to address these issues, we propose a sliding window accumulator structure, in which the hash of all data objects is calculated, with the result recorded as the root hash of the sliding window accumulator. Only these root hash values are stored on the blockchain.

4.3. Trustworthy Verification

4.3.1. DID Registry

During the DID register phase, it is necessary to associate the user's identity information with the verifiable registry. In the VS-DID model, we encode the user's identity information as a vector and compute the vector commitment to establish the binding relationship. Specifically, in the

Table 1
Symbols used in the DID Verifiable Registry

Symbol	Description
$\mathbf{W}$	Identity vector, $\mathbf{W} = (w_0, w_1, \ldots, w_{l-1})$.
w_i	i-th element of $\mathbf{W}$.
C	Commitment to $\mathbf{W}$.
π_i	Proof for w_i.
π_U	Aggregated proof for subset U.
C_U	Aggregated commitment for subset U.
g_1, g_2	Generators of $\mathbb{G}_1$ and $\mathbb{G}_2$.
e	Bilinear pairing, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
$\mathbb{G}_T$	Target group of bilinear pairing.
$\mathbf{A}$	Trapdoor in commitment scheme.
$Y_{k,n}$	Selection function in multilinear polynomial.
λ	Security parameter.
pp	Public parameters.
SK	Secret key.
PVK	Public verification key.
U	Subset of indices in $\mathbf{W}$.
Δ	Number of positions w_i is shifted.
$sup_{i,j}$	Supplementary info for updating proofs.
$x_{i,j}$	j-th node in binary tree path for w_i.
b	Number of identical bits between two paths.

DID register phase, the user generates a commitment C and corresponding proof π for their identity information vector $\mathbf{W}$ and then submits a registry request. The DID verifier then validates the proof using the provided commitment. Upon successful validation, the commitment and proof are recorded in the verifiable registry. To ensure efficient aggregation and management of commitments, we employ a binary linear tree structure as the foundational framework for the commitment scheme. Table 1 shows the symbols and descriptions usd in DID registry. The proposed vector commitment scheme includes four phases: generation of proof, aggregation of proof, verification of proof, and update of proof.

Generation of proof. We initially adopt the principles of Hyperproof [41] and polynomial commitments [52] by representing the original identity information vector $\mathbf{W} = (w_0, w_1, ..., w_{l-1}) \in \mathbb{Z}_p^l$ using a multilinear extended polynomial function f. Subsequently, we construct the commitment using a bilinear pairing group, where the group $\mathbb{G}_1$ and $\mathbb{G}_2$ are generated by the respective generators g_1 and g_2, and the pairing e mapping $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Specifically, let us denote $g_2^{t_j - i_j} (j \in [0, n])$ as the verification key located at the i-th position of $\mathbf{W}$.

The commitment is shown in Eq. (1).

$$C = g_1^{f(\mathbf{A})} = g_1^{\sum_{k=0}^{l-1} w_k Y_{k,n}(\mathbf{A})} = \prod_{j=0}^{n-1} (g_1^{Y_{k,n}(\mathbf{A})})^{w_k}, \quad (1)$$

where $\mathbf{A}$ denotes a trapdoor, $Y_{k,n}$ ($k \in [0,2^n)$) denotes the selection function. The trapdoor A is used to generate the commitment C. It ensures that the commitment is binding and hiding, meaning that an adversary cannot forge a valid commitment or extract information about the committed values without knowing the knowledge of A. To reduce the proof generation time, we employ a binary linear tree structure to recursively decompose the multilinear extended polynomial function. All commitments involved in the path to root of w_i represents the proof π_i for w_i. Specifically, the proof generation process consists of the following functions:

- $pp, SK, PVK \leftarrow \mathsf{Gen}(\lambda)$: Inputs the security parameter λ. Outputs the public parameters pp, public verification key PVK, and secret key SK.
- $C \leftarrow \mathsf{Com}(\mathbf{W})$: Inputs the identity information vector $\mathbf{W} = (w_0, w_1, ..., w_{l-1})$ and outputs the commitment C of $\mathbf{W} \in \mathbb{Z}_p^l$.
- $\pi_i \leftarrow \mathsf{Open}(w_i, i)$: Inputs the the position i and the element w_i at the i-th position in $\mathbf{W}$. The algorithm outputs the corresponding proof π_i.

Aggregation of proof. In the VS-DID model, we aggregate the proofs of individual elements w_i in the user's identity information vector $\mathbf{W}$ into a single aggregated proof. This aggregated proof verifies the integrity and accuracy of the entire vector $\mathbf{W}$. Inspired by Hyperproof [41], we apply the *Non-interactive Inner Product Arguments* (IPA) to facilitate proof aggregation. We use the IPA to generate an aggregated proof π_U and a corresponding aggregated commitment C_U from all individual proofs and their associated verification keys. Specifically, the proof aggregation process consists of the following function:

- $\pi_U, C_U \leftarrow \mathsf{Aggr}(w_i, \pi_i, U)$ $(i \in U)$: Inputs the individual vector element w_i and the corresponding proof π_i. The algorithm outputs the aggregated proof π_U and aggregated commitment C_U.

Verification of proof. During the verification phase, the public verification key is g_2, and the user submits the proof $\pi_i = (x_{i,n}, x_{i,n-1}, \ldots, x_{i,1})$, where $x_{i,n}$ is the n-th node of the i-th path in the binary linear tree. The DID verifier verifies the validity of the proof by computing $e(C/g_1^{a_i}, g_2)$ and $\prod_{j\in[0,n]} e(x_{i,j}, g_2^{a_j - i_j})$. When Eq. (2) holds, the proof is successfully verified.

$$e(C/g_1^{a_i}, g_2) = \prod_{j\in[0,n]} e(x_{i,j}, g_2^{a_j - i_j}). \quad (2)$$

Specifically, the proof verification process consists of the following function:

- $(0, 1) \leftarrow \mathsf{Verify}(C, U, \pi_U, (w_i)_{i\in U}, (PVK_i)_{i\in U})$: Inputs the proof π_U, the commitment C and the vector element w_i. The algorithm outputs 0 or 1.

Update of proof. To enhance the practicality and maintainability of VS-DID, we address proof updates when the positions of elements in the identity information vector are altered. Suppose the position of element w_i in vector $\mathbf{W}$ is shifted by Δ positions. To achieve the update of the

corresponding commitment and proof, supplementary information about element w_i, as outlined in Eq. (3), is required.

$$sup_{i'} = \left\{ sup_{i',j}, j \in [0,n] \right\} = \left\{ g_1^{Y_{i',j}(\mathbf{A})}, j \in [0,n] \right\}. \quad (3)$$

For any element w_i, its proof update after shifting by Δ positions is provided in Eq. (4).

$$x'_{i',j} = x_{i',j} \cdot (sup_{i',j-1})^{\Delta} = x_{i',j} \cdot (g_1^{Y_{i',j-1}(\mathbf{A})})^{\Delta}. \quad (4)$$

Furthermore, for both the pre-update proof π_i and post-update proof $\pi_{i'}$, we consider the same nodes between the i-th path and the i'-th path in the binary linear tree. Specifically, if indices i and i' share b identical bits, it implies that for any $j \in [n-b+1, n]$, $i_j = i'_j$. The proof π_i for w_i is updated as demonstrated in Eq. (5).

$$x'_{i,j} = x_{i,j} \cdot (sup_{i,j-1})^{\Delta} = x_{i,j} \cdot (g_1^{Y_{i,j-1}(\mathbf{A})})^{\Delta} \quad (5)$$

The commitment C is updated as follows:

$$C' = C \cdot g_1^{Y_{i',n}(\mathbf{A})} = C \cdot (sup_{i',n})^{\Delta}. \quad (6)$$

Specifically, the proof update process consists of the following function:

- $C', \pi' \leftarrow \mathsf{Update}(C, i', \Delta)$: Inputs the commitment C, the i'-th position, and the shifted position Δ. Outputs the updated commitment C' and proof π'.

Security of DID Registry. DID Register scheme meets the correctness and soundness of the classic commitment scheme. Brief observation are given as follow:

Theorem 4.1. ***Correctness*** *If an honest prover $\mathcal{P}$ follows the protocol to generate a commitment C and proof π_U for a subset $U \subseteq \{0, 1, \ldots, l-1\}$, then for any $i \in U$, the verifier $\mathcal{V}$, using the correct input, will always accept the proof when running* $\mathsf{Verify}(C, U, \pi_U, (w_i)_{i\in U}, (PVK_i)_{i\in U})$. *Formally,* ***Correctness*** *of DID registry can be quantified as follows:*

$$Pr\left[\mathsf{Verify}(C, i, \pi_i, w_i, PVK_i) = 1\right] = 1. \quad (7a)$$

$$Pr\left[\mathsf{Verify}(C, U, \pi_U, w_i, PVK_i) = 1\right] = 1. \quad (7b)$$

Observation. Given a security parameter λ, the public parameters pp, a vector $\mathbf{W} = (w_0, w_1, \ldots, w_{l-1}) \in \mathbb{Z}_p^l$, and the secret key pair (SK, PVK). $\mathcal{P}$ generates a commitment C to the vector $\mathbf{W}$ using $C \leftarrow \mathsf{Com}(\mathbf{W})$. Since $\mathcal{P}$ is honest, the commitment C correctly binds to $\mathbf{W}$. For each position $i \in U$, $\mathcal{P}$ uses the private key SK, which is known only to the honest prover, to generate a proof π_i for w_i at position i by $\pi_i \leftarrow \mathsf{Open}(w_i, i)$. $\mathcal{P}$ aggregates the individual proofs into a single proof π_U for the subset U using $\pi_U \leftarrow \mathsf{Aggr}((w_i, \pi_i)_{i\in U})$. The aggregated proof π_U is computed in such a way that it can be used to verify the correctness of the values $\{w_i\}_{i\in U}$ with respect to the commitment C. The verifier $\mathcal{V}$ receives $(C, U, \pi_U, (w_i)_{i\in U}, (PVK_i)_{i\in U})$. To check the validity of the proof, $\mathcal{V}$ runs the verification algorithm $b \leftarrow \mathsf{Verify}(C, U, \pi_U, (w_i)_{i\in U}, (PVK_i)_{i\in U})$, where $b \in \{0, 1\}$ indicates whether the proof is accepted ($b = 1$) or rejected ($b = 0$).

By the construction of the *DID Register*, if all steps are followed correctly by $\mathcal{P}$, then for any $i \in U$, the verification algorithm must return 1. This is due to the commitment C and the individual proofs π_i being generated in accordance with the protocol, and the aggregation π_U being consistent with these individual proofs.

Theorem 4.2. ***Soundness*** *The probability of finding two different messages generating the same commitment is negligible. Formally, for the different vectors V and U and the Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, the chance of winning the verifier is negligible:*

$$Pr\left[\begin{array}{c} pp, SK, PVK \leftarrow \mathsf{Gen}(\lambda), \\ \left(C, \begin{array}{c}(U, w_i, PVK_i, \pi_U)_{i\in U}, \\ (V, w'_j, PVK_j, \pi'_V)_{j\in V}\end{array}\right) \leftarrow \mathcal{A}(\lambda, pp) : \\ 1 \leftarrow \mathsf{Verify}\begin{pmatrix} C, U, \pi_U, \\ w_i, PVK_i \end{pmatrix}_{i\in U} \wedge \\ 1 \leftarrow \mathsf{Verify}\begin{pmatrix} C, V, \pi'_V, \\ w'_j, PVK_j \end{pmatrix}_{j\in V} \wedge \\ \exists t \in U \cap V, w_t \neq w'_t \end{array}\right] \leq \mathrm{negl}(\lambda). \quad (8)$$

Observation. It is computationally infeasible for adversary $\mathcal{A}$ to find two different vectors $\mathbf{W}_1 \neq \mathbf{W}_2$ such that: $C = \mathsf{Com}(\mathbf{W}_1) = \mathsf{Com}(\mathbf{W}_2)$.

Suppose $\mathcal{A}$ can find two different vectors $\mathbf{W}_1$ and $\mathbf{W}_2$ such that $(g^{a_0}, g^{a_1}, \ldots, g^{a_{l-1}}) = \mathsf{Com}(\mathbf{W}_1) = \mathsf{Com}(\mathbf{W}_2)$. This implies that $\mathcal{A}$ can find two different sets of exponents $(a_0, a_1, \ldots, a_{l-1})$ and $(a'_0, a'_1, \ldots, a'_{l-1})$ such that $g^{a_i} = g^{a'_i}, \forall i \in \{0, 1, \ldots, l-1\}$. Since g is a generator of a cyclic group of prime order p, only happen if $a_i \equiv a'_i \pmod p), \forall i \in \{0, 1, \ldots, l-1\}$. It means that the exponents a_i and a'_i must be the same modulo p.

Finding two different sets of exponents that produce the same commitment would imply that the adversary can break the q-SDH assumption. Specifically, if $\mathcal{A}$ could find two different sets of exponents that result in the same commitment, they could use this information to compute $g^{1/(x+c)}$ for some $c \in \mathbb{Z}_p^*$, which contradicts the q-SDH assumption [41]. Formally, the probability that a polynomial-time adversary can find such $\mathbf{W}_1$ and $\mathbf{W}_2$ is negligible:

$$\Pr[\exists \mathbf{W}_1 \neq \mathbf{W}_2 : C = \mathsf{Com}(\mathbf{W}_1) = \mathsf{Com}(\mathbf{W}_2)] \leq \mathrm{negl}(\lambda).$$

By leveraging **Correctness** and **Soundness**, our model ensures that DID documents are tamper-evident. The commitment C binds the vector $\mathbf{W}$ in DID document to a unique value, and any change to $\mathbf{W}$ would require recomputing C, which is impossible without the trapdoor $\mathbf{A}$. Specifically, **Correctness** ensures that legitimate DID documents are always verified. **Soundness** makes it computationally infeasible to forge proofs for tampered documents.

4.3.2. *Verifiable DID Query Processing*

Traditional blockchain query processing suffers from inefficiency and insufficient performance. In the worst case, the time required for a query is linearly related to the number of data objects in the block. Therefore, using inter-block aggregation queries can help improve query performance.

Whenever a user needs to query a series of behaviors, they can use aggregated data objects for Boolean range queries. For example, in three blocks, there are three data objects o_1, o_2, o_3. o_1 contains keywords {A, B}, o_2 contains keywords {B, C}, and o_3 contains keywords {C, D}. If the query q = { "C" ∩ "E" }, the three data objects can be aggregated to obtain the set $S = \{A, B, C, D\}$. The user can then easily prove that the query does not hold because { "C" ∩ {A, B, C, D} } = ∅ . However, this traditional method of inter-block aggregation has significant limitations. If query q = { "A" ∩ "D"}, following the previous inter-block aggregation method, query q satisfies the set S, causing the inter-block aggregation index not to work properly, thus degrading to linear query processing.

In the VS-DID model, when a user submits a series of requests to a verifier, the DID verifier first needs to verify the correctness of the user's DID and VC. After obtaining the user's DID and VC, the verifier queries the off-chain stored data information. The verifier extracts keywords from all query requests within a time range and divides them into several subqueries. Each subquery corresponds to a sliding window accumulator with a window size of K. For each subquery, the sliding window accumulator index is used to find the corresponding keyword object set. Then, the intersection operation is performed on the found keyword object sets to verify whether the query is valid. If the result of the intersection operation is an empty set, the query is deemed invalid; otherwise, the system returns the result of the intersection operation.

The sliding window accumulator is similar to a Merkle tree. It hashes several different block data objects to generate the root of a sliding window accumulator query tree. This data structure can significantly improve the performance of the blockchain. The mechanism creates a sliding window accumulator index for each data object on the most recent k blocks, where k is also the window size of the sliding window accumulator. During query processing, we first search the sliding window accumulator index to obtain the corresponding keyword object sets. Then, we use intersection operations to verify whether the query holds. For the previously mentioned example, $q = \}\}A\varepsilon \cap \}\}D\varepsilon$, we search the sliding window accumulator index and obtain the object sets where keywords A and D appear, which are $\{o_1\}$ and $\{o_3\}$ respectively. Then, we perform an intersection operation and get an empty set, successfully proving the query is invalid. The design of the sliding window accumulator solves the issue of query failures caused by traditional inter-block aggregation. It also effectively improves query efficiency and optimizes system performance.

Algorithm 1 introduces the implementation of Boolean queries. Given a query format $Q = ([t_s, t_e], \Upsilon)$, the system should return all results that satisfy the intersection with the query keywords. The algorithm can be executed in three steps: First, the query is divided into a series of subqueries, each corresponding to a sliding window of size k. Then, each subquery uses the sliding window accumulator index to get the results. Finally, all results are merged to generate the final result. The specific process of using the sliding window accumulator for Boolean queries can be divided into the following three phases.

Algorithm 1 Boolean Query Processing

Require: Query condition $Q = ([t_s, t_e], \Upsilon)$
Ensure: Query result R_T

```
R ← ∅; qs ← DivideQuery(Q)
for each q in qs do
    ([t'_s, t'_e], Υ) ← q
    R_trie ← ∅
    for each keyword w in Υ do
        R_w ← QuerySWATrie(w, b_e'.root)
        Add R_w to R_trie
    end for
    R_T ← BooleanComputation(R_trie, Υ)
end for
return R_T
```

Phase I: Divide Query Range. For a query Q, if the query's time window is larger than k, then Q will be divided into several subqueries with a size of k. If the time window of the last subquery is smaller than k, the last subquery's time window will overlap with the previous subquery. This may lead to redundant query results but will not affect the correctness of the final query.

Phase II: Subquery Processing. For each subquery $q = ([t_s, t_e], \Upsilon)$ with a time window of k, we first traverse the sliding window accumulator tree on block b_e to find the results of this subquery. This subquery result includes all database objects corresponding to the keywords. Then, we can perform Boolean expression operations on all database objects according to the query keyword requirements to get the result R_r.

Phase III: Subquery Result Aggregation. All subquery results are aggregated to obtain the final query result and the Merkle proof path.

Algorithm 2 introduces the implementation of a range query function. Since the query process in a two-dimensional time range is almost similar to Algorithm 1 in dividing the query range, this description will focus on subquery processing without detailing the process of dividing the query range under range queries.

Given a subquery $q = ([t_s, t_e], [\alpha, \beta])$, we first traverse the sliding window accumulator tree on block b_e. For the root node n of the subtree, if its range $[l_n, u_n]$ is completely covered by $[\alpha, \beta]$, the system directly generates the Merkle proof path. If the range $[l_n, u_n]$ partially intersects with $[\alpha, \beta]$, the comparison continues with the child nodes of the root node n until the root node's range is fully covered by $[\alpha, \beta]$.

Algorithm 2 Range Query Processing

Require: SWA-B Tree root, query condition $[\alpha, \beta]$
Ensure: Query result R

```
Initialize an empty queue; queue.enqueue(root)
while queue is not empty do
    n ← queue.dequeue()
    if [l_n, u_n] ⊆ [α, β] then
        R ← R_n
    else if [α, β] ∩ [l_n, u_n] ≠ ∅ then
        for each child n' of n do
            queue.enqueue(n')
        end for
    else
        R ← ∅
    end if
end while
return R
```

Algorithm 3 Verifiable Registry Update

Require: New key-value pairs $[k_j, v_j]_j$ at time i, current state st_i at time i
Ensure: New dictionary summary d_{i+1}, new state st_{i+1}

```
(AHD, d_o, st_o) ← Init(·)
for each [k_i, v_i] in the set of key-value pairs do
    Add [k_i, v_i] to AHD
end for
if VerificationLookup(d_i, [k_i, v_i], π_lookup) == 1 then
    Swap ([k_i, v_i], [k_j, v_j])
end if
return d_{i+1}, st_{i+1}
```

If the range $[l_n, u_n]$ does not intersect with $[\alpha, \beta]$ at all, the search directly returns a path proof failure.

4.3.3. DID Update

In practical IIoT applications, new users continuously join the system due to the implementation of meta-computing, which causes user information to change over time. Therefore, our model needs to consider the situation of user information updates. We describe an authenticated dictionary-based process of verifiable user information updates in the following.

During the identity issuance phase, the DID issuer pairs each user's DID with their public key and stores the generated key-value pair in a verifiable registry. This verifiable registry is based on an RSA authenticated dictionary, which maintains version immutability proof. This authenticated dictionary is an extension of KVaC authenticated dictionary structure. Key-Value commitment is a cryptographic technique used to bind key-value pairs to a commitment. It proves their existence and consistency without exposing the specific key-value contents. This technique has broad applications in privacy protection and data verification. It allows generating a commitment that includes a hash or other form of summary of the data without revealing the actual data content. Then, the user can publish this commitment at any time and, when needed, prove that the commitment contains specific key-value pairs by providing the corresponding key and value to verify that these key-value pairs are consistent with the original data.

In a KVaC-based authenticated dictionary, each key-value pair is bound to a commitment. This commitment can be a hash value or another form of summary of the key-value pair. This commitment can be made public for verifiers to check the existence and consistency of the data. When a specific key-value pair needs to be verified, the verifier can provide the corresponding key and value and compare them with the previously published commitment. If the provided key-value pair matches the content in the commitment, it proves that this key-value pair exists in the original data and was bound to the commitment at a certain time. In this paper, the verifiable registry is an extension of the KVaC authenticated dictionary, combined with the RSA encryption algorithm. Whenever a new user registers, the new key-value pair is paired with a commitment, ensuring its immutability and integrity. This commitment is essentially a summary. Whenever a user's DID or public key changes, the registry updates the key-value pair by first calculating the new value and summary, then pairing them, and updating the registry's version number to maintain version immutability. No two identical key-value pairs can exist under the same version number. As an extension of the KVaC authenticated dictionary, for stronger key-value binding, this paper uses an append-only version number to track all historical mapping records. Thus, this paper introduces the new cryptographic primitive of an authenticated historical dictionary to maintain the registry.

Algorithm 3 introduces the key-value update function in the verifiable registry. Whenever a key-value pair needs to be updated and replace the old key-value pair, it is necessary to verify the consistency of the key-value pairs at time i, the corresponding user proof paths, and the summary in the initialized authenticated dictionary. This prevents malicious tampering with the dictionary records. Only when the key-value pairs, summary, and user information are correct can the key-value pairs in the registry be updated. After updating the key-value pairs, it is necessary to regenerate the corresponding summary for the time period to ensure its integrity and immutability. The design and implementation of this verifiable registry ensure data integrity, immutability, and historical traceability in the distributed digital identity system by introducing key-value commitments, RSA authenticated dictionaries, and historical authenticated dictionaries. This design enhances system security and reliability and provides strong support for applications in other fields.

5. Experiment and the Results

5.1. Experiment Configuration

The experiment of VS-DID is conducted on a hardware environment consisting of an Intel(R) Core(TM) i5-1035G4 CPU @ 1.10GHz processor, 40GB of storage, and 4GB of RAM, running the Ubuntu 20.04 operating system. The

Table 2
Single-threaded Experiment Evaluation of DID Registry. *Update commitment time* and *Update all proofs time* is the total time costs for 1024 changes to the vector. For *Update all proofs time*, dividing the total time by 1024 gives an average time per update of 0.03ms to 12.50 ms.

Time Costs	n =30									
	w=4	w=8	w=16	w=32	w=64	w=128	w=256	w=512	w=1024	w=2048
Aggregate proofs time (s)	0.04	0.08	0.16	0.32	0.64	1.26	2.48	4.93	9.83	19.85
Verify an individual proof time (s)	0.02	0.03	0.06	0.12	0.24	0.47	0.94	1.89	3.77	7.66
Verify an aggregated proof time (s)	0.04	0.07	0.13	0.25	0.49	0.95	1.82	3.15	6.64	12.76
Update commitment time (s)	0.02	0.04	0.07	0.12	0.24	0.44	0.81	1.50	2.73	4.93
Update all proofs time (s)	0.03	0.07	0.14	0.26	0.50	0.94	1.84	3.48	6.65	12.79

Table 3
Our Merkle Tree-based Accumulator Scheme

Time	$n = 8$	$n = 16$	$n = 32$	$n = 64$
Parameter Generation (s)	0.257	0.305	0.340	0.510
Prover (s)	6.227	7.272	8.077	10.417
Verifier(s)	0.003	0.003	0.003	0.003

objective of the experiment is to evaluate the performance of DID registry and query mechanisms in VS-DID, focusing on vector commitment for DID registry and a sliding window accumulator for DID queries.

For the DID registry process, the vector commitment scheme is implemented in Golang using the `mcl` cryptographic library[1]. The `mcl` library is designed for efficient elliptic curve pairing operations, specifically supporting the BLS12-381 elliptic curve. It provides optimized operations for the algebraic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, making it highly suitable for cryptographic applications that rely on pairing-based schemes. In this experiment, the `mcl` library handles the algebraic operations required for generating and verifying vector commitments. To further improve efficiency, the IPA-based aggregation scheme from `Hyperproof`[2] is used, enabling the aggregation of multiple commitments and proofs into a single verification step, reducing computational overhead. The experimental setup includes different vector sizes $w \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$, and Merkle tree heights $n \in \{3, 4, 5, 30\}$, to examine their effect on the performance of the registry process.

For the DID query process, the sliding window accumulator scheme is implemented using the `Bellman-Bignat` cryptographic library[3]. This library is specialized for large integer operations and building SNARK circuits, which are essential for cryptographic constructs such as Merkle accumulators. In this experiment, the Merkle accumulator is constructed using the `Bellman-Bignat` library's large integer operations to enable efficient querying of DID sets. Additionally, zero-knowledge proof verification for the accumulator is implemented using the Groth16 proving system, with SNARK circuits built on the BLS12-381 elliptic curve, ensuring both cryptographic security and operational efficiency. The performance is evaluated with different vector sizes $w \in \{16, 32, 64, 128, 256, 512, 1024, 2048\}$.

[1] https://github.com/herumi/mcl.git
[2] https://github.com/hyperproofs/hyperproofs-go.git
[3] https://github.com/alex-ozdemir/bellman-bignat

5.2. Experiment Results

Overall Results. We observe that when $n = 30$, the size of the public parameters for the Merkle SNARK scheme reaches approximately 50GB, and the parameter generation process requires over 30 hours. Given the significant time costs associated with the SNARK-based commitment scheme, we choose to balance between the values of w and n. For our VS-DID scheme, we maintain the height of the bilinear tree at $n = 30$ and assess the time performance for $w \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024\}$. In contrast, for the Merkle SNARK scheme, we adjust the Merkle tree height between $n = 3$ and $n = 5$ for the vector sizes $w \in \{8, 16, 32, 64\}$.

In this experiment, we evaluate the performance of the vector commitment scheme for DID registry using several key metrics: *Aggregate Proofs Time*, *Individual Proof Verification Time*, *Aggregated Proof Verification Time*, *Commitment Update Time*, and the time to *Update All Proofs*. These metrics are chosen to assess the efficiency and scalability of the system across different vector sizes. *Aggregate Proofs Time* and *Aggregated Proof Verification Time* measure the efficiency of aggregating and verifying multiple proofs simultaneously, which is essential for handling large datasets. *Individual Proof Verification Time* reflects the system's performance when verifying smaller subsets of data, which is important for real-time applications. *Commitment Update Time* captures the system's responsiveness when updating the vector commitment due to changes in DID information, while update all proofs time evaluates the cost of maintaining proof consistency as vector sizes increase. These metrics provide insights into the system's scalability and efficiency in dynamic environments where frequent updates and verifications are required, helping us understand its potential for real-world applications.

The experiment results in Table 2 and Table 3 are obtained through empirical measurements in our experiments. In Table 2, the *Update Commitment Time* and *Update all proofs time* are calculated by executing 1024 update operations and recording the total time taken. The total time is then divided by 1024 to obtain the average time per update. Similarly, other values such as *Aggregate proofs time* costs

Table 4
Merkle SNARK scheme versus DID registry.

Time Costs	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 time (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 time (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 time costs (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

and *Verify time* costs are measured by recording the total time to generate and verify aggregated proofs, and then averaging over the number of operations. In Table 3, *Parameter Generation Time* is measured by recording the time taken to generate the parameters for the Merkle tree accumulator, *Prover Time* is calculated by measuring the time taken to generate a proof, and *Verifier Time* is the time taken to verify the generated proof, with the verification time being consistent across different values of n. All experiments are conducted on standard hardware and software configurations (detailed in Section 5.1), and each experiment is repeated multiple times to ensure consistency. The values reported in the tables are the averages of these repeated trials.

The experimental results for DID registry using vector commitments, as shown in Table 2, illustrate the performance for varying vector sizes w with a fixed Merkle tree height $n = 30$. As the vector size increases from $w = 4$ to $w = 2048$, the time costs for different operations exhibit a noticeable growth. Specifically, the time for aggregating proofs rises from 0.04 seconds at $w = 4$ to 19.85 seconds at $w = 2048$, while the time to verify an individual proof increases from 0.02 seconds to 7.66 seconds across the same range. Similarly, verifying an aggregated proof takes from 0.04 seconds at $w = 4$ to 12.76 seconds at $w = 2048$. The time required to update the commitment and all proofs also increases with the vector size, where the update commitment time ranges from 0.02 to 4.93 seconds, and the update all proofs time ranges from 0.03 to 12.79 seconds.

Table 3 presents the experimental results for the DID query process using a Merkle tree-based accumulator scheme. The results evaluate three key time metrics: parameter generation time, prover time, and verifier time, across different Merkle tree heights ($n = 8$, $n = 16$, $n = 32$, and $n = 64$). For parameter generation time, as the tree height increases, the time also grows. It starts at 0.257 seconds for $n = 8$, rising to 0.510 seconds for $n = 64$, indicating that higher tree depths lead to increased computational overhead during the generation of parameters. The prover time similarly increases with the tree height, starting at 6.227 seconds for $n = 8$ and reaching 10.417 seconds for $n = 64$. This trend shows that the time required for the prover to generate proofs becomes longer as the complexity of the Merkle tree grows. In contrast, the verifier time remains constant at 0.003 seconds across all tree heights. This demonstrates that the verification process is highly efficient and unaffected

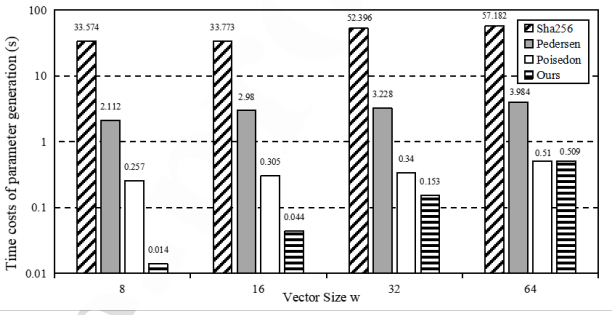

Figure 2: Parameter generation time of different algorithms under different vector sizes.

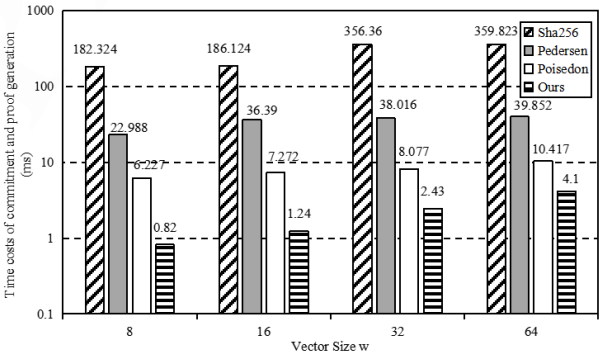

Figure 3: Commitment and proof generation time of different algorithms under different vector sizes.

by the increase in Merkle tree height, ensuring fast proof verification even as the underlying structure becomes more complex. Overall, the results highlight the scalability of the scheme, with parameter generation and prover times increasing with tree height, while verification remains constant and efficient.

In addition, we compared the effects of different vector size w, Sha256, Pedersen, Poisedon, and our scheme are at different stages of time. Fig. 2 shows the time costs of parameter generation, Fig. 3 shows the time costs of commitment and proof generation, and Fig. 4 shows the proof verification time. The experimental results demonstrate the superiority of our proposed method in terms of efficiency, scalability,

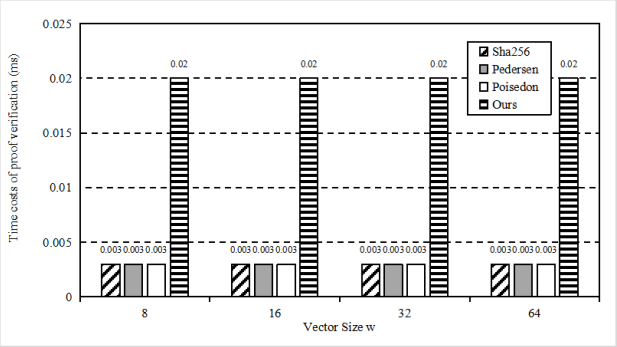

Figure 4: Proof verification time of different algorithms under different vector sizes.

and security. Specifically, our method achieves significantly lower parameter generation times (Fig. 2), more efficient commitment and proof generation (Fig. 3), and consistently low proof verification times (Fig. 4) compared to traditional approaches. These results indicate that our method not only reduces computational overhead but also scales effectively with larger vector sizes, making it highly suitable for DID verifiable authentication in meta-computing-enabled IIoT environments. By leveraging advanced cryptographic techniques such as vector commitments and Merkle accumulators, our approach ensures robust security while maintaining high performance, outperforming existing methods in key metrics.

Compared with Merkle SNARK. The experimental results presented in Table 4 compare the time performance of the Merkle SNARK scheme and our DID registry scheme for various operations across different Merkle tree heights n and vector size w. The table evaluates three key metrics: aggregate proofs time, verification time, and commitment time costs. For aggregate proofs time, the Merkle SNARK scheme shows increasing time costs as both n and w grow. For example, at $w = 8$ and $n = 3$, it takes 0.32 seconds, and as n increases to 5, this grows to 0.44 seconds. Similarly, at $w = 64$, the time increases from 2.54 seconds ($n = 3$) to 3.44 seconds ($n = 5$). In contrast, our scheme demonstrates much lower aggregate proof times across all configurations, with times ranging from 0.02 seconds at $n = 3$ and $w = 8$ to 0.14 seconds at $n = 5$ and $w = 64$, indicating significantly better efficiency compared to Merkle SNARK. For verification time, the Merkle SNARK scheme maintains a constant verification time of 0.003 seconds across all configurations of n and w, showing no performance degradation as the tree height or window size increases. Our scheme, on the other hand, shows slightly higher verification times, starting at 0.01 seconds at $n = 3$ and $w = 8$, and rising to 0.06 seconds at $n = 5$ and $w = 64$. While our scheme has higher verification times, it remains efficient and scales reasonably with larger parameters. Finally, for commitment time costs, the Merkle SNARK scheme incurs significantly higher time costs, especially as n and w increase. For instance, at $w = 8$ and $n = 3$, the time is 11.88 seconds, but at $w = 64$ and $n = 5$, it jumps to 138.17 seconds. In comparison, our scheme exhibits much lower commitment times, starting at 0.65 seconds for $n = 3$ and $w = 8$, and increasing to 5.51 seconds for $n = 5$ and $w = 64$, reflecting a substantial improvement in performance over the Merkle SNARK scheme.

Overall, the results demonstrate that while both schemes handle verification efficiently, our DID registry scheme significantly outperforms the Merkle SNARK in terms of aggregate proofs and commitment time, making it a more scalable and practical solution for large-scale applications.

In large-scale user and data environments, the DID verifiable registry is proposed to efficiently handle secure identity verification requests and potential performance bottlenecks. Specifically, the parameters used for proof generation in the DID verifiable registry can be reused, which significantly reduces computational overhead and supports efficient identity verification for a large number of users. As shown in Fig. 4, the proof verification time for DID registry remains consistently low at 0.003 seconds, ensuring real-time responsiveness even under large-scale verification requests. Furthermore, the experimental results in Table 2 and Table 4 demonstrate that our model supports efficient proof aggregation and updates. Proof aggregation allows multiple verification requests to be processed in batches, significantly reducing the overall verification time. This capability enables VS-DID to handle DID verification requests efficiently, even as the number of users and data scales up. By combining reusable parameters, low verification times, and efficient proof aggregation, our model effectively addresses the challenges of scalability in large-scale environments.

6. Conclusion

This paper addressed the issue of authentication in meta-computing-enabled IIoT and proposed a verifiable computation-based DID scheme for ensuring both secure scalable storage and authentications. The proposed model adopted a sliding window accumulator to facilitate boolean and range queries. We also utilized key-value commitments in a verifiable registry for trustworthy queries within the authentication mechanism. Our model had been demonstrated to be effective in guaranteeing data integrity and tampering-resistant.

For future work, we plan to further explore the scalability of our model in large-scale industrial applications by integrating it with other emerging technologies such as edge computing and blockchain. Additionally, we aim to improve the performance of our authentication mechanism by optimizing the key-value commitment scheme for reduced overhead in large systems. Finally, we will investigate the impact of different cryptographic techniques on the overall security and efficiency of the system, especially in high-latency or resource-constrained environments.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (Grant No. 2021YFB2701300),

National Natural Science Foundation of China (Grant No.s U24B20146, 62372044), and Beijing Municipal Science and Technology Commission Project (Z241100009124008).

References

[1] Adeniyi, E.A., Jimoh, R.G., Awotunde, J.B., 2024. A systematic review on elliptic curve cryptography algorithm for internet of things: Categorization, application areas, and security. Computers and Electrical Engineering 118, 109330.

[2] Ahmed, M.R., Islam, A.M., Shatabda, S., Islam, S., 2022. Blockchain-based identity management system and self-sovereign identity ecosystem: A comprehensive survey. IEEE Access 10, 113436–113481.

[3] Alquraan, A., Udayashankar, S., Marathe, V.J., Wong, B., Al-Kiswany, S., 2024. Lolkv: The logless, linearizable, rdma-based key-value storage system, in: 21st USENIX Symposium on Networked Systems Design and Implementation, Santa Clara, CA. pp. 41–54.

[4] Alsadi, M., Casey, M., Dragan, C.C., Dupressoir, F., Riley, L., Sallal, M., Schneider, S., Treharne, H., Wadsworth, J., Wright, P., 2023. Towards end-to-end verifiable online voting: Adding verifiability to established voting systems. IEEE Transactions on Dependable and Secure Computing 21, 3357–3374.

[5] Ansari, M.N., Razaq, A., Alolaiyan, H., Shuaib, U., Salman, M.A., Xin, Q., 2024. Empowering decentralized identity systems for web 3.0 in complex spherical fuzzy knowledge. Scientific Reports 14, 23590.

[6] Baldimtsi, F., Karantaidou, I., Raghuraman, S., 2024. Oblivious accumulators, in: IACR International Conference on Public-Key Cryptography, Springer. pp. 99–131.

[7] Bao, H., Zhang, X., Wang, G., Tian, R., Duan, J., Zhao, Y., 2023. Smart-pki: A blockchain-based distributed identity validation scheme for iot devices, in: ICC 2023-IEEE International Conference on Communications, IEEE. pp. 4749–4754.

[8] Barthoulot, A., Blazy, O., Canard, S., 2024. Cryptographic accumulators: new definitions, enhanced security, and delegatable proofs, in: International Conference on Cryptology in Africa, Springer, Douala, Cameroon. pp. 94–119.

[9] Cai, T., Chen, W., Psannis, K.E., Goudos, S.K., Yu, Y., Zheng, Z., Wan, S., 2022. Scalable on-chain and off-chain blockchain for sharing economy in large-scale wireless networks. IEEE Wireless Communications 29, 32–38.

[10] de Castro, L., Lee, K., 2024. Verisimplepir: Verifiability in simplepir at no online cost for honest servers , 5931–5948.

[11] Chen, B., Waiwitlikhit, S., Stoica, I., Kang, D., 2024. 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. pp. 560–574.

[12] Cheng, X., Xu, M., Pan, R., Yu, D., Wang, C., Xiao, X., Lyu, W., 2023. Meta computing. IEEE Network 38, 225–231.

[13] Cremonezi, B., Vieira, A.B., Nacif, J., Silva, E.F., Nogueira, M., 2024. Identity management for internet of things: concepts, challenges and opportunities. Computer communications PP, 99.

[14] Deebak, B.D., Hwang, S.O., 2024. Privacy preserving based on seamless authentication with provable key verification using miomt for b5g-enabled healthcare systems. IEEE Transactions on Services Computing 17, 1097–1113.

[15] Dib, O., Rababah, B., 2020. Decentralized identity systems: Architecture, challenges, solutions and future directions. Annals of Emerging Technologies in Computing 4, 19–40.

[16] Dushku, E., Rabbani, M.M., Vliegen, J., Braeken, A., Mentens, N., 2023. Prove: Provable remote attestation for public verifiability. Journal of Information Security and Applications 75, 103448.

[17] Fotiou, N., Siris, V.A., Polyzos, G.C., Kortesniemi, Y., Lagutin, D., 2022. Capabilities-based access control for iot devices using verifiable credentials, in: 2022 IEEE Security and Privacy Workshops (SPW), IEEE, San Francisco, CA, USA. pp. 222–228.

[18] Gai, K., Wu, Y., Zhu, L., Qiu, M., Shen, M., 2019a. Privacy-preserving energy trading using consortium blockchain in smart grid. IEEE Trans. on Industrial Informatics 15, 3548–3558.

[19] Gai, K., Wu, Y., Zhu, L., Zhang, Z., Qiu, M., 2019b. Differential privacy-based blockchain for industrial internet-of-things. IEEE Trans. on Industrial Informatics 16, 4156–4165.

[20] Gai, K., Zhang, Y., Qiu, M., Thuraisingham, B., 2022. Blockchain-enabled service optimizations in supply chain digital twin. IEEE Transactions on Services Computing 16, 1673–1685.

[21] Ge, C., Liu, Z., Susilo, W., Fang, L., Wang, H., 2023. Attribute-based encryption with reliable outsourced decryption in cloud computing using smart contract. IEEE Transactions on Dependable and Secure Computing 21, 937–948.

[22] Hao, J., Huang, C., Tang, W., Zhang, Y., Yuan, S., 2021. Smart contract-based access control through off-chain signature and on-chain evaluation. IEEE Transactions on Circuits and Systems II: Express Briefs 69, 2221–2225.

[23] Jonathan, R., Henning, S., Gonzalo, C., Alan, J., iothers, 2002. SIP: session initiation protocol. Technical Report.

[24] Kara, M., Merzeh, H.R., Aydın, M.A., Balık, H.H., 2023. VoIPChain: A decentralized identity authentication in voice over IP using blockchain. Computer Communications 198, 247–261.

[25] Karaja, M., Chaabani, A., Azzouz, A., Ben Said, L., 2023. Dynamic bag-of-tasks scheduling problem in a heterogeneous multi-cloud environment: a taxonomy and a new bi-level multi-follower modeling. The Journal of Supercomputing 79, 17716–17753.

[26] Khan, A.A., Wagan, A.A., Laghari, A.A., Gilal, A.R., Aziz, I.A., Talpur, B.A., 2022. Biomt: A state-of-the-art consortium serverless network architecture for healthcare system using blockchain smart contracts. IEEE Access 10, 78887–78898.

[27] Krawczyk, H., Paterson, K.G., Wee, H., 2013. On the security of the tls protocol: A systematic analysis, in: Annual Cryptology Conference, Springer, Santa Barbara, CA, USA. pp. 429–448.

[28] Li, Y., Guo, J., Li, Y., Wang, T., Jia, W., 2023. An online resource scheduling for maximizing quality-of-experience in meta computing. arXiv preprint arXiv:2304.13463 .

[29] Liu, C., Guo, H., Xu, M., Wang, S., Yu, D., Yu, J., Cheng, X., 2022. Extending on-chain trust to off-chain–trustworthy blockchain data collection using trusted execution environment (tee). IEEE Transactions on Computers 71, 3268–3280.

[30] Liu, H., Han, D., Cui, M., Li, K.C., Souri, A., Shojafar, M., 2023. Idenmultisig: Identity-based decentralized multi-signature in internet of things. IEEE Transactions on Computational Social Systems 10, 1711–1721.

[31] Liu, Q., Peng, Y., Xu, M., Jiang, H., Wu, J., Wang, T., Peng, T., Wang, G., 2024. MPV: enabling fine-grained query authentication in hybrid-storage blockchain. IEEE Transactions on Knowledge and Data Engineering 36, 3297–3311.

[32] Luecking, M., Fries, C., Lamberti, R., Stork, W., 2020. Decentralized identity and trust management framework for internet of things, in: 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE. pp. 1–9.

[33] Maram, D., Malvai, H., Zhang, F., Jean-Louis, N., Frolov, A., Kell, T., Lobban, T., Moy, C., Juels, A., Miller, A., 2021. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability, in: 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA. pp. 1348–1366.

[34] Mazzocca, C., Acar, A., Uluagac, S., Montanari, R., 2024. Evoke: Efficient revocation of verifiable credentials in iot networks, in: 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA. pp. 1279–1295.

[35] Parameswarath, R.P., Gope, P., Sikdar, B., 2023. Privacy-preserving user-centric authentication protocol for iot-enabled vehicular charging system using decentralized identity. IEEE Internet of Things Journal 6, 70–75.

[36] S.Ali, 2020. Distributed ledger technology. Internet computing: Principles of distributed systems and emerging internet-based technologies pp, 265–299.

[37] Sallal, M., de Fréin, R., Malik, A., 2023. Pvpbc: Privacy and verifiability preserving e-voting based on permissioned blockchain. Future Internet 15, 121.

[38] Sedlmeir, J., Smethurst, R., Rieger, A., Fridgen, G., 2021. Digital identities and verifiable credentials. Business & Information Systems Engineering 63, 603–613.
[39] Shen, X., Luo, X., Yuan, F., Wang, B., Chen, Y., Tang, D., Gao, L., 2024. Verifiable privacy-preserving federated learning under multiple encrypted keys. IEEE Internet of Things Journal 11, 3430–3445.
[40] Smarr, L., Catlett, C.E., 1992. Metacomputing. Communications of the ACM 35, 44–52.
[41] Srinivasan, S., Chepurnoy, A., Papamanthou, C., Tomescu, A., Zhang, Y., 2022. Hyperproofs: Aggregating and maintaining proofs in vector commitments, in: 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA. pp. 3001–3018.
[42] S.Si, X.Lin, P.Wei, 2024. Compressed zero-knowledge proofs for lattice-based accumulator. The Computer Journal 67, 694–708.
[43] Szalachowski, P., 2021. Password-authenticated decentralized identities. IEEE Transactions on Information Forensics and Security 16, 4801–4810.
[44] W3C, 2018. Decentralized Identifiers v0.11: Data Model and Syntaxes for Decentralized Identifiers. Technical Report.
[45] Wang, J., Hu, J., Min, G., Zomaya, A.Y., Georgalas, N., 2020. Fast adaptive task offloading in edge computing based on meta reinforcement learning. IEEE Transactions on Parallel and Distributed Systems 32, 242–253.
[46] Wang, L., Tian, Y., Zhang, D., 2021. Toward cross-domain dynamic accumulator authentication based on blockchain in internet of things. IEEE Transactions on Industrial Informatics 18, 2858–2867.
[47] Wang, X., Cao, J., Buyya, R., 2022. Adaptive cloud bundle provisioning and multi-workflow scheduling via coalition reinforcement learning. IEEE Transactions on Computers 72, 1041–1054.
[48] Xie, T., Gai, K., Zhu, L., Guo, Y., Choo, K.K.R., 2023. Cross-chain-based trustworthy node identity governance in internet of things. IEEE Internet of Things Journal 10, 21580–21594.
[49] Xin, J., Haghighi, A., Tian, X., Papadopoulos, D., 2024. Notus: Dynamic proofs of liabilities from zero-knowledge {RSA} accumulators, in: 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA. pp. 1453–1470.
[50] Xiong, R., Ren, W., Hao, X., He, J., Choo, K.K.R., 2023. Bdim: A blockchain-based decentralized identity management scheme for large scale internet of things. IEEE Internet of Things Journal 10, 22581–22590.
[51] Xu, C., Zhang, C., Xu, J., Pei, J., 2021. Slimchain: Scaling blockchain transactions through off-chain storage and parallel processing. Proceedings of the VLDB Endowment 14, 2314–2326.
[52] Zhang, L.F., Wang, H., 2022. Multi-server verifiable computation of low-degree polynomials, in: 2022 IEEE Symposium on Security and Privacy (SP), IEEE, San Francisco, CA, USA. pp. 596–613.
[53] Zhang, Z., Yang, K., Tian, Y., Ma, J., 2024. An anti-disguise authentication system using the first impression of avatar in metaverse. IEEE Transactions on Information Forensics and Security 19, 6393–6408.
[54] Zhuang, T., Qin, Z., Ding, Y., Deng, F., Chen, L., Qin, Z., Choo, K.K.R., 2023. Temporal refinement graph convolutional network for skeleton-based action recognition. IEEE Transactions on Artificial Intelligence 5, 1586–1598.