Security and Safety, Vol. 3, 2024013 (2024)
https://doi.org/10.1051/sands/2024013
sands.edpsciences.org

Research Article OPEN ACCESS

Digital Finance

Supervised and revocable decentralized identity privacy protection scheme

Jing He[1], Xiaofeng Ma[1,*], Dawei Zhang[2,3], and Feng Peng[4]

[1] *School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China*
[2] *School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*
[3] *Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China*
[4] *China Securities Information Technology Service Limited Company, Beijing 100033, China*

Received: 19 August 2024 / Revised: 23 September 2024 / Accepted: 15 October 2024 / Published online: 30 October 2024

Abstract Decentralized identity represents an innovative approach based on blockchain to achieve effective identity management. This method utilizes decentralized identifiers and verifiable credentials to enable trusted authentication, free circulation of identity information, and self-sovereign control over identity data functionalities. The current decentralized identity systems rely on entirely anonymous identifiers, lacking robust identity regulation. Furthermore, they face challenges such as identity attribute leakage during verifiable credential presentation and the issuers' struggle to reliably revoke credentials. To address these issues, efficient and practical schemes have been designed based on BBS signature, zero-knowledge proof, dynamic accumulator, and blockchain technology: one for decentralized identifiers management and the other for verifiable credential privacy protection, both of which are supervised and revocable. The former ensures the privacy of subject identity while achieving regulatability and revocability of identity data by the regulator. The latter facilitates selective disclosure of anonymous credentials and reliable revocation. A security analysis shows that the proposed scheme meets anonymity, non-forgeability, regulatory reliability, and revocability reliability, and offers comprehensive and effective privacy protection measures. The experimental results demonstrate that the algorithms designed operate at a millisecond level, which satisfies the demands of blockchain identity management scenarios.

Keywords Blockchain, Decentralized identity, Privacy protection, Supervision, Anonymous credential, Zero-knowledge proof

Citation He J, Ma X and Zhang D et al. Supervised and revocable decentralized identity privacy protection scheme. Security and Safety 2024; **3**: 2024013. https://doi.org/10.1051/sands/2024013

1 Introduction

In the contemporary era of pervasive digitalization and intricate networking, digital identity has become a fundamental component for individuals and organizations to interact and access services in cyberspace. One important application scenario is regional equity markets, which facilitate the issuance, trading, and management of equity shares within specific geographic regions. These markets, as an integral part of China's multi-tiered capital market system, are characterized by decentralization, relatively low transaction volumes, and stringent requirements for compliance and security. Currently, China has 35

* Corresponding author (email: xiaofengma@tongji.edu.cn)

regional equity markets, primarily serving small and medium-sized enterprises and offering financing solutions for "specialized, refined, differentiated, and innovative" enterprises. In this environment, regulatory authorities play a key role in supervising identity verification and transaction processes to ensure market integrity and legal compliance. To balance privacy protection with regulatory oversight, robust identity and credential management systems are essential.

In recent years, the distributed digital identity framework proposed by the World Wide Web Consortium (W3C) has attracted wide attention. Based on the blockchain infrastructure, this framework provides a distributed, transparent, and secure method for identity management. It is designed to enhance users' control over their personal data and significantly mitigate the risk of single points of failure and the potential for data tampering [1]. However, distributed digital identity systems also face several issues and challenges in practical applications, particularly concerning regulatory adaptability, privacy protection, and technical implementation.

Distributed digital identity systems utilize decentralized identifiers (DIDs) to enhance user privacy and anonymity. However, this approach introduces significant regulatory challenges. In the absence of explicit identity information, regulators encounter difficulties in tracking and auditing transactions, thereby compromising the legitimacy and security of financial operations [2]. Furthermore, verifiable credentials (VCs) in these systems often contain multiple attributes. During the verification process, it is often sufficient to ascertain a subset of attributes in VCs. However, there exists a non-negligible risk that the system may inadvertently disclose additional sensitive information, which can lead to privacy-compromising and an increased risk of identity theft. While some systems offer selective disclosure of attributes, the continued use of the same DIDs or the lack of support for randomized signatures can result in linkability issues, where multiple interactions are tied to the same user. This undermines privacy protection [3]. Additionally, the reliable revocation of credentials is crucial to prevent unauthorized access or misuse. However, existing solutions are characterized by an absence of robust and efficient mechanisms designed to manage the revocation process in a manner that ensures both security and scalability.

This paper introduces an innovative solution to address the regulatory and privacy challenges in decentralized identity management through two tightly integrated schemes: SR-DIDs (Supervised and Revocable Decentralized Identifiers) and SR-PP-VC (Supervised and Revocable Privacy-Preserving Verifiable Credentials). The SR-DIDs scheme facilitates the derivation of master DIDs based on Pedersen's commitment, which enables users to autonomously create anonymous proxy DIDs while maintaining their privacy. A key innovation lies in the integration of BBS signatures, DLIN encryption, and zero-knowledge proofs, which not only empowers regulatory authorities to discern user identities when necessary but also ensures a secure and reliable revocation of DIDs through smart contracts. The proposed scheme ensures anonymity, unforgeability, and regulatory compliance.

The SR-PP-VC scheme builds on SR-DIDs by extending its regulatory and privacy-preserving features to verifiable credentials. By leveraging BBS signatures and dynamic accumulators, it facilitates real-time revocation of credentials and empowers users to present their credentials in a manner that ensures unlinkability. The tight integration between SR-DIDs and SR-PP-VC ensures coherence in both identity and credential management, thereby providing a unified framework where DIDs and VCs are both regulated and revocable in a harmonious manner. Selective disclosure of credential attributes via zero-knowledge proofs further strengthens privacy, addressing issues of over-disclosure prevalent in current systems.

We implemented the system on Hyperledger Fabric and conducted comprehensive simulations to validate the proposed approach. The results demonstrate that our solution not only enhances security and privacy but also significantly improves scalability and efficiency compared to existing systems. The presented framework meets the performance and security demands of blockchain-based identity management, providing a novel and effective solution to the challenges of privacy protection, regulatory oversight, and credential revocation.

2 Related work

At present, the mainstream blockchain uses identity identifiers or digital certificates to define the ownership of assets, data, and rights [4]. For example, the public chain relies on anonymous addresses to identify the identity, and the alliance chain confirms the identity of members through real-name digital certificates.

To prevent identifier association from causing user identity disclosure [5], researchers proposed to derive one-time identifiers from public and private keys through encryption algorithms to reduce the correlation between identifiers and enhance identity privacy, such as Monero [6], literature [7], literature [8] and other schemes. In order to further implement the supervision of one-time identifiers, literature [9] designed a link ring signature algorithm, which realized the privacy protection scheme of transaction sender and receiver identifiers that could be regulated by the autonomous confusion mechanism. However, the ring signature size was very large, resulting in large performance consumption. Literature [10] designed a supervised one-time identifier generation scheme to hide the real identity of the transaction receiver but did not support identifier revocation.

To solve the problem that the public and transparent feature of blockchain leads to the leakage of users' real identity information when digital certificates are presented, Hyperledger Fabric uses Idemix solution [11], which can realize the anonymity and unlinkability of users' identities and the selective disclosure of certificate attributes. Document [12] uses a state secret SM2 signature to create anonymous credentials to ensure user anonymity. In order to solve the certificate supervision problem, literature [13] designed a blockchain-oriented regulatory anonymous certificate authentication scheme based on the BBS04 group signature [14] and Idemix scheme but did not support certificate revocation. Literature [15] constructs a traceable and randomizable anonymous certificate based on the PS signature, but the certificate size increases linearly with attributes. As for the revocation of digital certificates, literature [16] builds an anonymous certificate that can be revoked, unlinked, and selectively disclosed based on a dynamic accumulator and PS signature algorithm. However, due to the fixed number of certificates issued, it is difficult to apply in practice and does not support identity supervision. Document [17] revocation certificates based on time fragments, but it is not a real-time operation and requires frequent updating of certificates, which consumes large storage resources.

To effectively manage a large amount of identity information, distributed digital identity technology integrates identity identifiers and credentials, builds the identity management layer of the blockchain system, and gives subjects control over their own identity data [4]. However, the research in this field is still in the initial stage, mainly focusing on its application practice. For example, the Internet of Things [18], medical health [19], finance [20], 6G communication [21], and other fields have effectively solved the problems faced by centralized digital identity systems, such as the lack of identity sovereignty, the disadvantages of centralized authentication, and large-scale identity identifiers and certificate management problems, and promoted the circulation and authentication of user identities among different organizations. Regarding the privacy protection of distributed digital identities, literature [22] proposes a VC privacy protection scheme with optional disclosure based on an editable signature algorithm. Literature [23] builds a new distributed digital identity system by combining a prophecy machine and trusted execution environment, which can obtain and issue identity credentials from Web service providers securely and conveniently, and protect user identity privacy. Literature [24] puts the VC generation process into a trusted execution environment to protect user attribute privacy. However, the above schemes do not consider identity supervision and VC revocation in the distributed digital identity system.

3 Propaedeutics

3.1 Blockchain

Blockchain is a decentralized and distributed ledger technology, first introduced in 2008 by an anonymous individual or group known as Satoshi Nakamoto, with the invention of Bitcoin [25]. The key feature of blockchain is its ability to securely record transactions across multiple computers, ensuring data integrity and transparency. The technical architecture of blockchain consists of several layers, including peer-to-peer networks, consensus mechanisms, cryptographic hash functions, and distributed ledgers. Each block contains a list of transactions that are cryptographically linked, ensuring immutability. Consensus mechanisms, such as Proof of Work (PoW) or Proof of Stake (PoS), ensure that all participants in the network agree on the validity of transactions, providing security and trust in a decentralized system.

One of the most important innovations in blockchain technology is the introduction of smart contracts, first implemented on the Ethereum blockchain. Smart contracts allow the automatic execution of contract terms on the blockchain without the need for intermediaries, enabling the automation of complex processes. This provides a flexible and powerful environment for developing and deploying decentralized

applications. Smart contracts have expanded blockchain applications into areas such as finance (DeFi), supply chain management, and healthcare, making blockchain a versatile tool for ensuring transparency, efficiency, and trust in decentralized environments.

In the legal field, Sharifi introduced a formal contract specification language called Symboleo, which encodes obligations and rights into smart contracts [26]. Domestically, Wang Di proposed SPESC, which translates legal contracts into smart contract languages to enhance their legal validity [27]. Zhu Yan further designed rules for converting SPESC into Solidity, a high-level smart contract language [28]. To address the issue of smart contract performance, Zhang Fuli developed a microservice framework for smart contracts by decoupling consensus nodes from smart contracts [29]. Liu proposed parallel execution of transactions by execution nodes and asynchronous transaction ordering by consensus nodes, enabling parallel execution of smart contracts [30]. Wang Zikai introduced the SBFT consensus algorithm and a task-parallel smart contract model, optimizing the efficiency and scalability of smart contract execution [31].

3.2 Bilinear map

Given that $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of prime order p, if there exists a mapping e such that e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, satisfying the following properties, then this mapping is called a bilinear map:

(1) **Bilinearity**: For any $\alpha, \beta \in \mathbb{Z}_p$, $g_1 \leftarrow \mathbb{G}_1$, $g_2 \leftarrow \mathbb{G}_2$, it holds that $e(g_1^{\alpha}, g_2^{\beta}) = e(g_1, g_2)^{\alpha\beta}$.

(2) **Non-degeneracy**: There exist $g_1 \leftarrow \mathbb{G}_1$, $g_2 \leftarrow \mathbb{G}_2$ such that $e(g_1, g_2) \neq 1$.

(3) **Computability**: For any $g_1 \leftarrow \mathbb{G}_1$, $g_1 \leftarrow \mathbb{G}_1$, there exists an efficient algorithm to compute $e(g_1, g_2)$.

Reference [32] defines three types of bilinear maps: **Type 1**: $\mathbb{G}_1 = \mathbb{G}_2$, also known as symmetric pairing. **Type 2**: $\mathbb{G}_1 \neq \mathbb{G}_2$, but there exists an efficient homomorphic mapping $\varnothing : \mathbb{G}_2 \to \mathbb{G}_1$. **Type 3**: $\mathbb{G}_1 \neq \mathbb{G}_2$, and there does not exist an efficient homomorphic mapping from $\mathbb{G}_2$ to $\mathbb{G}_1$. The scheme proposed in this paper is based on Type 3 bilinear maps.

3.3 Cryptographic assumptions

(1) q-DL Assumption

Given that $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic groups of order p, with g_1 and g_2 being generators of groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, it is computationally hard to solve for the integer x from the given (q+3)-tuple $(g_1, g_1^x, g_1^{x^2}, \ldots, g_1^{x^q}, g_2, g_2^x)$.

(2) q-SDH Assumption

Given $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic groups of order p, with g_1 and g_2 being generators of groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, it is computationally hard to output the pair $(g_1^{1/(x+c)}, c)$ from the given (q+3)-tuple $(g_1, g_1^x, g_1^{x^2}, \ldots, g_1^{x^q}, g_2, g_2^x)$, where $x, c \in \mathbb{Z}_p$.

(3) DLIN Assumption

Given $\mathbb{G}_1$ is a cyclic group of order p it is computationally hard to determine if $c = a + b$ holds from a set of elements U, V, W, U^a, V^b, W^c, where U, V, W are random elements in group $\mathbb{G}_1$ and $a, b, c \in \mathbb{Z}_p$ are randomly chosen.

3.4 BBS signature

The BBS signature scheme was initially part of the BBS04 group signature scheme [14] and was explicitly defined as an independent signature scheme in literature [33]. The original BBS signature lacked security proof until 2023 when it was shown in the literature [34] that the BBS signature satisfies strong existential unforgeability under chosen message attack (SUF-CMA) under the q-SDH assumption. The BBS signature process is as follows:

(1) $Setup(1^{\lambda})$: Input the system parameter 1^{λ} to generate the Type 3 bilinear map $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. Let g_1 and g_2 be the generators of groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Randomly select a group element $h_1 \leftarrow \mathbb{G}_1$. Output the public parameters $pp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, h_1, g_2)$.

(2) $KeyGen(pp)$: Randomly choose $x \in \mathbb{Z}_p$ as the private key sk. Compute the public key $pk = g_2^x$.

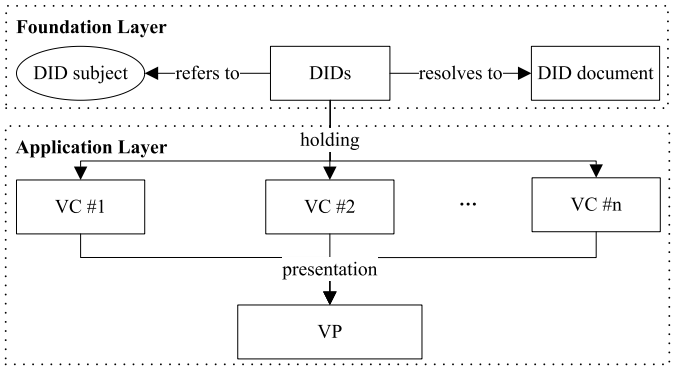


Figure 1. Basic component relationships of decentralized identity

(3) $Sign(m, sk, pp)$: Input the message $m \in \mathbb{Z}_p$ and the private key sk. Randomly select an integer $k \in \mathbb{Z}_p$ and compute $A = (g_1 \cdot h_1^m)^{1/(x+k)})$. Output the signature $\sigma = (A, k)$.

(4) $Verify(\sigma, m, pp)$: Verify if $e(A, g_2^k \cdot pk) = e(g_1 \cdot h_1^m,\ g_2)$. If the equation holds, the signature is valid; otherwise, it is invalid.

3.5 Distributed digital identity

Distributed digital identity leverages blockchain technology to transform the traditional centralized identity control model into a distributed control model. It establishes an identity management layer for blockchain systems, enabling entities to own and manage their identity data. The distributed digital identity scheme designed in this paper follows the W3C standards, with its basic component structure illustrated in Figure 1.

(1) Foundation Layer: DIDs serve as unique addresses for each entity, linked to detailed DID documents. These documents contain the entity's public key information, authentication protocols, and service endpoints, facilitating multi-dimensional, flexible expression of identity and context-adaptive privacy protection.

(2) Application Layer: VCs are digitally issued certificates by authoritative entities, recording the attributes and qualifications of the entity in various contexts. Entities are allowed to independently hold and verify multiple identity attributes. VPs are composite proof structures built on VCs, enabling entities to provide comprehensive, cryptographically signed proofs of identity to third parties by selecting and combining multiple VCs.

4 Architecture design

In the regional equity market scenario, the participants–regulatory authorities, issuers, investors, and trading platforms–play distinct roles. The regulatory authority is assumed to be a trustworthy and secure entity, overseeing identity, transaction, and credential management to ensure compliance and market safety. Issuers and trading platforms are considered credible and legitimate institutions that closely collaborate with regulators and adhere to market rules. On the other hand, investors, whose identities and activities must be verified through credentials, are not regarded as fully trusted entities. Against this backdrop, the scheme is designed to protect identity and credential privacy while allowing regulatory oversight to ensure legal and transparent financial activities. Thus, the scheme does not pursue full decentralization but instead enhances system transparency and security through blockchain technology, striking a balance between regulatory oversight and privacy protection to ensure market transparency and safety.

This paper classifies DIDs into two categories, the overall architecture of digital identity management in this paper is illustrated in Figure 2.

(1) **Main Decentralized Identifiers (MDIDs)**: Users register with regulatory authorities using real-name information. Upon approval by the regulatory authority, users can possess and use an MDID, with each user limited to one MDID.

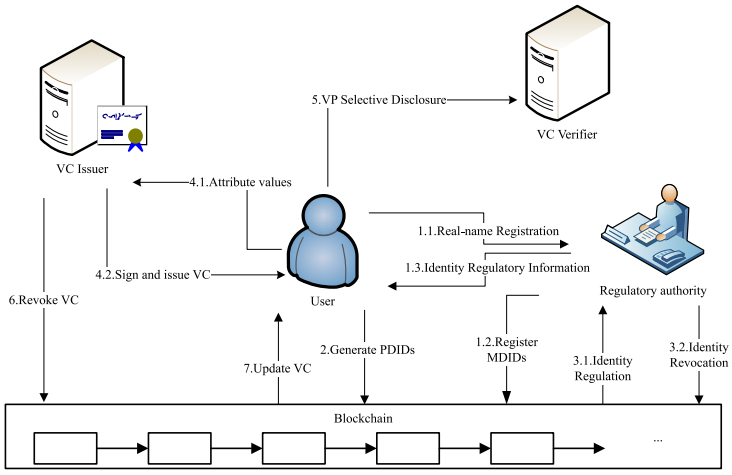


Figure 2. Architecture of identity management

(2) **Proxy Decentralized Identifiers (PDIDs):** Users can independently generate multiple PDIDs. In this paper, only the regulatory authority can associate PDIDs with MDIDs.
The specific process is as follows:

(1) **Real-name Registration**: Users undergo real-name authentication with the regulatory authority. Upon successful authentication, the regulatory authority registers an MDID for the user via a smart contract and signs the MDID with their private key, serving as the user's identity regulatory key.

(2) **PDID Generation**: Based on their MDIDs and identity regulatory key, users independently generate unique PDIDs and corresponding regulatory information, uploading them to the blockchain for automatic verification by the smart contract.

(3) **Dispute or Violation Resolution**: In case of disputes or violations, the regulatory authority uses the regulatory information of PDIDs to trace the user's real identity and revoke the corresponding DIDs if necessary.

(4) **VC Application**: Users apply for VCs using any DIDs and their identity attributes. The issuer verifies the attributes and issues the VCs.

(5) **VC Verification**: Users randomize and combine their VCs into a VP as required by the verifier, hiding unnecessary attribute values. The verifier validates the VP's signature to determine its validity.

(6) **VC Revocation**: The VC issuer can update the accumulator and the revocation list on the blockchain, thereby revoking a specific VC.

(7) **Credential Updates**: Users check the latest revocation list and update their credentials accordingly, ensuring that revoked credentials are no longer usable.

SR-DIDs and SR-PP-VC schemes are designed to work in tandem to achieve regulatory oversight and revocability of decentralized identities (DIDs) and verifiable credentials (VCs) within the system. The SR-DIDs scheme focuses on the management of decentralized identifiers, ensuring that user identities remain anonymous while providing regulatory authorities the ability to recover the true identity through regulatory information when necessary. It allows for the creation of proxy DIDs, which can be traced back to the main DIDs under certain regulatory conditions, ensuring both privacy and accountability.

On the other hand, the SR-PP-VC scheme extends the regulatory and revocation capabilities to the verifiable credentials associated with these DIDs. By utilizing BBS signatures and dynamic accumulators, the SR-PP-VC scheme enables selective disclosure of credential attributes while ensuring that VCs can be reliably revoked in real-time. The two schemes complement each other: SR-DIDs manage the identity layer by handling DIDs, and SR-PP-VC manages the credential layer by ensuring the privacy, regulatory compliance, and revocation of credentials. Together, they provide a cohesive framework for privacy protection and regulatory oversight in decentralized identity systems.

5 Regulated and revocable DIDs management scheme

5.1 Scheme definition

The SR-DIDs scheme comprises eight algorithms, described as follows:

(1) **Public System Parameter Generation** $SetUp(1^\lambda) \rightarrow pp$: Input the system security parameter 1^λ. Output the system public parameters pp, which include the DIDs generation base DR used for constructing entity DIDs.

(2) **Regulator Key Pair Generation** $RegulatorKeyGen(pp) \rightarrow (RSK, RPK, DIDsRL)$: Executed by the regulator, input the public parameters pp. Output the regulator's private key RSK, public key RPK, and DIDs revocation list $DIDsRL$.

(3) **MDIDs Request Generation** $MDIDsReqGen(pp) \rightarrow (UMDIDsReq, umsk, d_m)$: Executed by the users, input the public parameters pp, output MDIDs-registration request $UMDIDsReq$, user's master secret key $umsk$, and user's MDID-associated key d_m. The $UMDIDsReq$ includes the main identity identifier $UMDIDs$ to be registered.

(4) **MDIDs Registration** $MDIDsReg(pp, MDIDsReq, RSK) \rightarrow (USK)$: Executed by the regulator, input the public parameters pp, the user's MDID registration request $MDIDsReq$, and the regulator's private key RSK. Output the user's identity regulatory key USK.

(5) **PDIDs Generation** $UserPDIDsGen(pp, umsk, d_p, RPK, USK, UMDIDs) \rightarrow (UPDIDs, UPProof)$: Executed by the users. Input the public parameters pp, the user's master secret key $umsk$, the user's PDID-associated key d_p, the regulator's public key RPK, the user's identity regulatory key USK, and the user's main decentralized identity identifier $UMDIDs$. Output the user's proxy decentralized identity identifier $UPDIDs$ and its regulatory information $UPProof$.

(6) **User PDIDs Verification** $UserPDIDsVer(pp, RPK, UPDIDs, UPProof) \rightarrow (0/1)$: Executed by the smart contract automatically, input the public parameters pp, the regulator's public key RPK, the user's proxy decentralized identity identifier $UPDIDs$, and its regulatory information $UPProof$. Output 1 if the verification passes; 0 otherwise.

(7) **Identity Supervision** $Supervise(UPDIDs, UPProof, RSK) \rightarrow (USK)$: Executed by the regulator, input the user's $UPDIDs$, its regulatory information $UPProof$, and the regulator's private key RSK. Output the identity regulatory key USK corresponding to the PDIDs.

(8) **Identity Revocation** $DIDsRevoke(RevDIDs) \rightarrow (DIDsRL)$: Executed by the regulator, input the identifiers to be revoked $RevDIDs$. Update the DID revocation list $DIDsRL$ to render $RevDIDs$ invalid within the system.

5.2 Security model

The security properties of the SR-DIDs scheme include anonymity, unforgeability, regulatory reliability, and revocation reliability. This scheme uses the following lists and oracles to describe its security properties:

$\mathcal{L}_{H\_U}, \mathcal{L}_{C\_U}$: Lists of honest and corrupt users, respectively, recording the indices i of honest and corrupt users. $\mathcal{L}_{H\_U} \cap \mathcal{L}_{C\_U} = \varnothing$.

$\mathcal{L}_{PDIDs}, \mathcal{L}_{d\_p}$: Lists of PDIDs and their corresponding PDID keys, recording the users' PDIDs and the keys associated with those PDIDs. For example, $\mathcal{L}_{PDIDs}[i]$ denotes the set of PDIDs owned by user i. $\mathcal{L}_{d\_p}$ corresponds one-to-one with each element in $\mathcal{L}_{PDIDs}$.

Oracle for Honest Users $\mathcal{O}_{H\_U}(i)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i to add a new user i. If $i \in \mathcal{L}_{H\_U} \cup \mathcal{L}_{C\_U}$, return $\perp$. Otherwise, execute $MDIDsReqGen(pp, umsk[i], d_m[i]) \rightarrow (MDIDsReq[i])$ and update $\mathcal{L}_{H\_U} = \mathcal{L}_{H\_U} \cup \{i\}$, then return $\top$.

Oracle for Corrupt Users $\mathcal{O}_{C\_U}(i)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i to obtain the user's identity information. If $i \notin \mathcal{L}_{H\_U} \cup \mathcal{L}_{C\_U}$, return $\perp$. If $i \in \mathcal{L}_{H\_U}$, update $\mathcal{L}_{H\_U} = \mathcal{L}_{H\_U} - \{i\}$ and $\mathcal{L}_{C\_U} = \mathcal{L}_{C\_U} \cup \{i\}$. Then return $USK[i], MDIDsReq[i],\ umsk[i], d_m[i], \mathcal{L}_{PDIDs}[i], \mathcal{L}_{d\_p}[i]$.

Oracle for MDIDs $\mathcal{O}_{MDIDs}(i)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i to obtain the identity regulatory key signed by the regulator for the user's MDIDs. If $i \notin \mathcal{L}_{H\_U} \cup \mathcal{L}_{C\_U}$, return $\perp$. Otherwise, execute $MDIDsReg(pp, MDIDsReq[i], RSK[i]) \rightarrow (USK[i])$. If $i \in \mathcal{L}_{H\_U}$, return $\top$. If $i \in \mathcal{L}_{C\_U}$, return $USK[i]$.

Oracle for PDIDs $\mathcal{O}_{PDIDs}(i)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i to generate the PDIDs and their regulatory information for user i. If $i \notin \mathcal{L}_{H\_U} \cup \mathcal{L}_{C\_U}$, return $\perp$.

Otherwise, execute $UserPDIDsGen(pp, d_p, RPK, USK[i], UMDIDs[i]) \rightarrow (UPDIDs, UPProof)$ and update $(\mathcal{L}_{PDIDs}[i],\ \mathcal{L}_{d\_p}[i]) = (\mathcal{L}_{PDIDs}[i], \mathcal{L}_{d\_p}[i])$. If $i \in \mathcal{L}_{C\_U}$, return $\top$. If $i \in \mathcal{L}_{C\_U}$, return $UPDIDs, UPProof$.

Supervision Oracle $\mathcal{O}_{Sup}(UPDIDs, UPProof)$: Adversary $\mathcal{A}$ queries this oracle with $UPDIDs$ and their regulatory information $UPProof$ to obtain the user's identity regulatory key. If $UPDIDs \notin \mathcal{L}_{PDIDs}$, return $\perp$. Otherwise, execute $Supervise(UPDIDs, UPProof, RSK) \rightarrow (USK[i])$ and return $USK[i]$.

Oracle for Public Information $\mathcal{O}_W$: Adversary $\mathcal{A}$ can query this oracle to obtain all public MDIDs, PDIDs, and their regulatory information from the blockchain system.

(1) Anonymity

In the SR-DIDs scheme, anonymity is defined as: except for the regulator, no other entities in the system can infer a user's identity information through DIDs or the regulatory information of PDIDs. Therefore, the anonymity of the SR-DIDs scheme also ensures the unlinkability of DIDs.

Definition 1. For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, if $\mathcal{A}$ can make a polynomially bounded number of queries to the oracle list $\mathcal{O} = \{\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{MDIDs}, \mathcal{O}_{PDIDs}, \mathcal{O}_W\}$, and the probability of Equation (1) is negligible, then the SR-DIDs scheme satisfies anonymity.

$$Pr\left[\begin{array}{c} SetUp\left(1^{\lambda}\right) \rightarrow pp; \\ RegulatorKeyGen(pp) \rightarrow (RSK, RPK, DIDsRL); \\ \mathcal{A}^{\mathcal{O}}(pp, RPK, DIDsRL) \rightarrow \{i_0, i_1\}; \\ \mathcal{C}\left(pp, i_b\right) \rightarrow \left(UMDIDs\left[i_b\right], UPDIDs_{i_b}, UPProof_{i_b}\right); \\ \mathcal{A}\left(pp, UMDIDs\left[i_b\right], UPDIDs_{i_b}, UPProof_{i_b}\right) \rightarrow b^* \end{array} : b^* = b\right] - \frac{1}{2} \tag{1}$$

(2) Unforgeability

In the SR-DIDs scheme, unforgeability is defined as: without the regulator's private key, no attacker can generate verifiable PDIDs and their regulatory information.

Definition 2. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can make a polynomially bounded number of queries to the oracle list $\mathcal{O} = \{\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{MDIDs}, \mathcal{O}_{PDIDs}, \mathcal{O}_W\}$, and the probability of Equation (2) is negligible, then the SR-DIDs scheme satisfies unforgeability.

$$Pr\left[\begin{array}{c} SetUp\left(1^{\lambda}\right) \rightarrow pp; \\ RegulatorKeyGen(pp) \rightarrow (RSK, RPK, DIDsRL); \\ \mathcal{A}^{\mathcal{O}}(pp, RPK, DIDsRL) \rightarrow (UPDIDs^*, UPProof^*); \\ UserPDIDsVer\left(pp, RPK, UPDIDs^*, UPProof^*\right) \rightarrow b^* : \\ (b^* = 1) \wedge (i \notin \mathcal{L}_{C\_U}) \wedge (UPDIDs^* \notin \mathcal{L}_{PDIDs}) \end{array}\right] \tag{2}$$

(3) Regulatory Reliability

In the SR-DIDs scheme, regulatory reliability is defined as: except for the regulator, even if users collude, no attacker can construct regulatory information that can pass system verification while preventing the regulator from tracing the real identity information. Therefore, the regulatory reliability of the SR-DIDs scheme also ensures traceability, for users operating honestly, the regulator can recover the user's regulatory key from the PDIDs' regulatory information.

Definition 3. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can make a polynomially bounded number of queries to the oracle list $\mathcal{O} = \{\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{MDIDs}, \mathcal{O}_{PDIDs}, \mathcal{O}_{Sup}, \mathcal{O}_W\}$, and the probability of Equation (3) is negligible, then the SR-DIDs scheme satisfies regulatory reliability.

$$Pr\left[\begin{array}{c} SetUp\left(1^{\lambda}\right) \rightarrow pp; \\ RegulatorKeyGen(pp) \rightarrow (RSK, RPK, DIDsRL); \\ \mathcal{A}^{\mathcal{O}}(pp, RPK, DIDsRL) \rightarrow (UPDIDs^*, UPProof^*); \\ UserPDIDsVer\left(pp, RPK, UPDIDs^*, UPProof^*\right) \rightarrow b^* : \\ (b^* = 1) \wedge (Supervise\left(UPDIDs^*, UPProof^*, RSK\right) \neq URSK[i]) \end{array}\right] \tag{3}$$

(4) Revocation Reliability

In the SR-DIDs scheme, revocation reliability is defined as: if the regulator chooses to revoke a particular DID, that DID can no longer be used, and attackers in the system cannot prevent the regulator from revoking DIDs or reactivating revoked DIDs.

5.3 Scheme design

(1) $SetUp(1^\lambda) \rightarrow pp$: First, input the system parameter 1^λ to generate a Type 3 bilinear map $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ and the integer group $\mathbb{Z}_p$. Choose the generators g_1 and g_2 for groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Then, randomly select two different group elements $DSK, DRand \leftarrow \mathbb{G}_1$, and set the DID generation base as $DR = (DSK, DRand)$. Also, select a cryptographic hash function $\mathcal{H} : \{0,\ 1\}^* \rightarrow \{0,\ 1\}^*$ that is one-way and collision-resistant. Finally, output the public parameters $pp = (p, \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T, \mathbb{Z}_p, g_1, g_2, e, \mathcal{H}, DR)$.

(2) $RegulatorKeyGen(pp) \rightarrow (RSK, RPK, DIDsRL)$: The regulator first selects an integer $y \in \mathbb{Z}_p$ and computes $\overline{Y} = g_2^y$. Then, select integers $u, v \in \mathbb{Z}_p$, and let $U = g_1^u$, $V = g_1^v$, and $W = U^v$, which implies $W = U^v = V^u = g_1^{uv}$. Set the regulator's private key as (y, u, v) and the public key as $(\overline{Y}, U, V, W)$. The private key is stored locally, while the public key is made public on the blockchain. Finally, initialize the DID revocation list $DIDsRL$ as an empty list and upload it to the blockchain.

(3) $MDIDsReqGen(pp) \rightarrow (MDIDsReq, umsk, d_m)$: The user first selects an integer $umsk \in \mathbb{Z}_p$ as their master private key. Then, select an MDID key $d_m \in \mathbb{Z}_p$ and generate the MDID based on the Pedersen commitment: $UMDIDs = DSK^{umsk} \cdot DRand^{d_m}$. Finally, integrate the user's real-name identity information $info$ into the registration request $UMDIDsReq = (UMDIDs, info)$ and send it to the regulator.

(4) $MDIDsReg(pp, MDIDsReq, RSK) \rightarrow (USK)$: Upon receiving the user's registration request $UMDIDsReq$, the regulator first verifies the $info$ and checks whether the $UMDIDs$ has already been registered in the system. If it is already registered, the application is invalid. If verified correctly, the regulator selects an integer $k_u \in \mathbb{Z}_p$ and signs the user's MDIDs with a BBS signature: compute $K = (g_1 \cdot UMDIDs)^{1/(y+k_u)}$. Finally, send $USK = (k_u, K)$ to the user as their identity regulatory key, and store $info$, $UMDIDs$, and $USK = (k_u, K)$) in the regulator's private table $RSList$. The $UMDIDs$ are then activated on the blockchain.

(5) $UserPDIDsGen(pp, d_p, RPK, USK, UMDIDs) \rightarrow (UPDIDs, UPProof)$: Users independently generate PDIDs based on their identity regulatory key to prevent other entities from tracking and linking.

① The user randomly selects $r_1, r_2 \in \mathbb{Z}_p$ and encrypts the identity regulatory key using DLIN: $R_1 = U^{1/r_1}$, $R_2 = V^{r_2}$, $R_3 = K \cdot W^{1/r_1+r_2}$. Then, randomize the calculation $\overline{K} = K^{r_1}$, and $\overline{J} = \overline{K}^{-k_u} \cdot (g_1 \cdot UMDIDs)^{r_1}$.

② Select a PDID key $d_p \in \mathbb{Z}_p$ and compute the PDID based on the Pedersen commitment: $UPDIDs = DSK^{umsk} \cdot DRand^{d_p}$. Calculate $t_1 = d_p - d_m$ and $t_2 = k_u / r_1$.

③ The user generates a zero-knowledge proof of knowledge for the correct generation of MDIDs, identity regulatory key, and PDIDs: $\pi_1 = NIZK - PoK_1\{r_1, r_2, t_1, t_2 | R_1 = U^{1/r_1} \wedge R_2 = V^{r_2} \wedge R_3 = \overline{K}^{1/r_1} \cdot W^{1/r_1+r_2} \wedge \overline{J} = (\overline{K}^{-t_2} \cdot g_1 \cdot UPDIDs \cdot DRand^{-t_1})^{r_1}\}$

a. Randomly select $\overline{r}_1, \overline{r}_2, \overline{r}_{t_1}, \overline{r}_{t_2} \in \mathbb{Z}_p$ and compute auxiliary values for the zero-knowledge proof: $R_1' = U^{\overline{r}_1}$, $R_2' = V^{\overline{r}_2}$, $R_3' = \overline{K}^{\overline{r}_1} \cdot W^{\overline{r}_1+\overline{r}_2}$, and $T_1' = \overline{J}^{\overline{r}_1} \cdot \overline{K}^{\overline{r}_{t_2}} \cdot DRand^{\overline{r}_{t_1}}$.

b. Compute the challenge value: $c_1 = \mathcal{H}(\overline{K}, \overline{J}, UPDIDs, R_1, R_2, R_3, R_1', R_2', R_3', T_1')$.

c. Compute the s-values: $s_{r_1} = \overline{r}_1 + c_1/r_1$, $s_{r_2} = \overline{r}_2 + c_1 \cdot r_2$, $s_{t_1} = \overline{r}_{t_1} + c_1 \cdot t_1$, and $s_{t_2} = \overline{r}_{t_2} + c_1 \cdot t_2$.

d. Generate the zero-knowledge proof $\pi_1 = (s_{r_1}, s_{r_2}, s_{t_1}, s_{t_2}, c_1)$.

④ Set the regulatory information for the user's PDIDs as $UPProof = (\overline{K}, \overline{J}, R_1, R_2, R_3, \pi_1)$, and the user uploads $(UPDIDs, UPProof)$ to the blockchain system.

(6) $UserPDIDsVer(pp, RPK, UPDIDs, UPProof) \rightarrow (0/1)$: In the blockchain system, the user-generated PDIDs can be treated as a transaction, which is automatically verified by a smart contract.

① Check if $UPDIDs$ already exist on the blockchain. If it does, the registration is rejected. Otherwise, verify if $e(\overline{K}, \overline{Y}) = e(\overline{J}, g_2)$. If it does not hold, the application is invalid. If it holds, proceed to the next verification step.

② Compute auxiliary values for verification: $R_1'' = U^{s_{r_1}} \cdot R_1^{-c_1}$, $R_2'' = V^{s_{r_2}} \cdot R_2^{-c_1}$, $R_3'' = \overline{K}^{s_{r_1}} \cdot W^{s_{r_1}+s_{r_2}} \cdot {R_3}^{-c_1}$, and $T_1'' = \overline{J}^{s_{r_1}} \cdot \overline{K}^{s_{t_2}} \cdot DRand^{s_{t_1}} \cdot (g_1 \cdot UPDIDs)^{-c_1}$.

③ Based on the public parameters and auxiliary values, compute the challenge value: $c_1' = \mathcal{H}(\overline{K}, \overline{J}, UPDIDs, R_1, R_2, R_3, R_1'', R_2'', R_3'', T_1'')$. Then verify if $c_1' = c_1$. If it holds, the application is approved and $UPDIDs$ is activated, outputting 1; otherwise, it is rejected, outputting 0.

(7) $Supervise(UPDIDs, UPProof, RSK) \rightarrow (USK)$: The regulator can access the PDIDs information $(UPDIDs, UPProof)$ from the blockchain at any time, then perform DLIN decryption: $K' =$

$R_3/(R_1^v \cdot R_2^u)$ to obtain one of the user's identity regulatory keys K'. The regulator compares it with their database $RSList$ to retrieve $USK, info$ and $UMDIDs$, thereby obtaining the real identity of the user applying for the PDIDs. The $UPDIDs$ is then associated with the user's $UMDIDs$ and added to the $RSList$. Subsequently, the regulator can directly query the user's real identity through $UPDIDs$.

(8) $DIDsRevoke(RevDIDs) \rightarrow (DIDsRL)$: If a particular DID $RevDIDs$ needs to be revoked, the regulator updates the DID revocation list $DIDsRL = DIDsRL \cup \{RevDIDs\}$ via a smart contract, rendering $RevDIDs$ invalid.

5.4 Security analysis

(1) Anonymity

Theorem 1. If the DLIN encryption scheme satisfies indistinguishability under chosen plaintext attacks (IND-CPA), the Fiat-Shamir non-interactive zero-knowledge (NIZK) proof protocol satisfies zero-knowledge, and the Pedersen commitment scheme satisfies hiding, then the SR-DIDs scheme satisfies anonymity.

Proof. If an adversary $\mathcal{A}$ can successfully attack the anonymity of the SR-DIDs scheme with non-negligible probability, then it is possible to construct a PPT algorithm $\mathcal{B}$ that can attack the IND-CPA property of the DLIN encryption scheme, the zero-knowledge property of the Fiat-Shamir NIZK proof protocol, or the hiding property of the Pedersen commitment scheme with the same probability. The interaction between $\mathcal{A}$ and $\mathcal{B}$ is as follows:

① **Setup Phase**: $\mathcal{B}$ simulates the public parameters of the SR-DIDs scheme based on the IND-CPA property of the DLIN encryption scheme, the zero-knowledge property of the Fiat-Shamir NIZK proof protocol, and the hiding property of the Pedersen commitment scheme. $\mathcal{B}$ constructs the system parameters pp and the regulator's public key RPK, and sends (pp, RPK) to $\mathcal{A}$.

② **Query Phase I**: $\mathcal{A}$ can query $\mathcal{B}$ for q times. $\mathcal{B}$ follows the procedures of the oracles $\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}$, $\mathcal{O}_{MDIDs}, \mathcal{O}_{PDIDs}, \mathcal{O}_W$ and returns the results to $\mathcal{A}$. During the DID generation and zero-knowledge proof generation, $\mathcal{B}$ obtains results from the simulated IND-CPA, zero-knowledge, and hiding games.

③ **Challenge Phase**: $\mathcal{A}$ selects two users i_0 and i_1 who have never been queried before and sends them to $\mathcal{B}$. $\mathcal{B}$ forwards them to the simulated IND-CPA, zero-knowledge, and hiding games and obtains the results $(UMDIDs[i_b], UPDIDs_{i_b}, UPProof_{i_b})$, which are sent to $\mathcal{A}$.

④ **Query Phase II:** $\mathcal{A}$ continues querying as in Query Phase I, but cannot query the oracle $\mathcal{O}_{C\_U}$ for users i_0 and i_1.

⑤ **Guessing Phase**: $\mathcal{A}$ guesses $\mathcal{B}$ and outputs b'. $\mathcal{B}$ forwards b' to the simulated IND-CPA, zero-knowledge, and hiding games.

If an attacker can successfully attack the IND-CPA property of the DLIN encryption scheme, the zero-knowledge property of the Fiat-Shamir NIZK proof protocol, or the hiding property of the Pedersen commitment scheme, then $\mathcal{A}$ can successfully attack the anonymity of the SR-DIDs scheme. However, since the DLIN encryption scheme satisfies IND-CPA, the Fiat-Shamir NIZK proof protocol satisfies zero-knowledge, and the Pedersen commitment scheme satisfies hiding, the advantage of $\mathcal{A}$ is negligible, and the probability in Equation (1) is negligible. Thus, the SR-DIDs scheme satisfies anonymity.

(2) Unforgeability

Theorem 2. If the q-SDH assumption holds and the zero-knowledge proof π_1 satisfies zero-knowledge, completeness, and soundness, then the SR-DIDs scheme satisfies unforgeability.

Proof. Assume $\mathcal{A}$ is an adversary of the unforgeability of the SR-DIDs scheme, $\mathcal{B}$ is an algorithm that attacks the q-SDH assumption or the zero-knowledge, completeness, and soundness of the Fiat-Shamir NIZK proof protocol, and $\mathcal{C}$ is the challenger of the q-SDH assumption simulation game, while $\mathcal{S}$ is the simulator for generating the zero-knowledge proof π_1 [35]. The interaction between $\mathcal{A}$ and $\mathcal{B}$ is as follows:

① **Setup Phase**: $\mathcal{C}$ sends the public parameters $(g_1, g_2, \overline{Y} = g_2^y)$ to $\mathcal{B}$. Then, $\mathcal{B}$ constructs (pp, RPK) based on the public parameters of the q-SDH simulation game and sends them to $\mathcal{A}$, retaining (u, v).

② **Query Phase**: $\mathcal{A}$ can query $\mathcal{B}$ for q times. $\mathcal{B}$ follows the procedures of the oracles $\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{MDIDs}, \mathcal{O}_{PDIDs}, \mathcal{O}_W$. For parts involving signatures, $\mathcal{B}$ constructs the commitment value to be signed and sends it to $\mathcal{C}$ for signing. For parts involving the zero-knowledge proof π_1, $\mathcal{S}$ runs the simulation. $\mathcal{B}$ returns the results to $\mathcal{A}$.

③ **Forgery Phase**: $\mathcal{A}$ constructs $(UPDIDs^*, UPProof^*)$ for an honest user i and sends them to $\mathcal{B}$.

④ **Judgment Phase**: If $(UPDIDs^*, UPProof^*)$ passes verification, it indicates $\mathcal{A}$ has successfully attacked. $\mathcal{B}$ then forwards $(\overline{K}^*, \overline{J}^*)$ to $\mathcal{C}$ and π_1^* to $\mathcal{S}$.

Since $\mathcal{A}$ and $\mathcal{B}$ are both PPT algorithms, the probability of $\mathcal{A}$ successfully attacking the unforgeability of the SR-DIDs scheme is the same as the probability of $\mathcal{B}$ successfully attacking the q-SDH assumption or the zero-knowledge, completeness, and soundness of the Fiat-Shamir NIZK proof protocol. However, the q-SDH assumption holds, and the Fiat-Shamir NIZK proof protocol satisfies zero knowledge, completeness, and soundness, so the advantage of $\mathcal{A}$ is negligible, and the probability in Equation (2) is negligible. Thus, the SR-DIDs scheme satisfies unforgeability.

(3) Regulatory Reliability

Theorem 3. If the SR-DIDs scheme satisfies unforgeability and regulatory correctness, then the SR-DIDs scheme satisfies regulatory reliability.

Proof. According to the regulatory reliability process defined in (3), if the regulator running the *Supervise* algorithm obtains a user identity regulatory key USK that does not belong to a corrupted user or has not appeared before, it means that the adversary $\mathcal{A}$ can break the unforgeability of the SR-DIDs scheme and forge the regulator's signature and π_1 to pass the blockchain consensus node verification. However, since the SR-DIDs scheme satisfies unforgeability, the probability of $\mathcal{A}$ breaking the unforgeability of the SR-DIDs scheme is negligible. Additionally, from the algorithm $(USK) \leftarrow Supervise(UPDIDs, UPProof, RSK)$, it is known that the regulator can recover the identity regulatory key from the PDIDs' regulatory information, satisfying traceability. Therefore, the probability in Equation (3) is negligible, and the SR-DIDs scheme satisfies regulatory reliability.

(4) Revocation Reliability

The SR-DIDs scheme uses blockchain technology for DID revocation, and the blockchain ensures tamper-resistance and transparency. If the status of DID changes, the ledger of all storage nodes on the blockchain will be updated, and all users will know that the DID has been revoked. Attackers cannot modify the ledger state or prevent the regulator from revoking DIDs unless they can compromise 1/3 of the nodes (if the blockchain uses the PBFT consensus algorithm) or 1/2 of the nodes (if the blockchain uses the Raft consensus algorithm), which is difficult to achieve in practice. Therefore, the SR-DIDs scheme satisfies revocation reliability.

6 Regulatorily accountable and revocable VC privacy protection scheme

6.1 Scheme definition

The SR-PP-VC scheme consists of eight main algorithms, described as follows:

(1) **Public System Parameter Generation** $SetUp(1^\lambda) \rightarrow pp$: Input the system security parameter 1^λ and output the system public parameters pp.

(2) **VC Issuer Key Generation** $VCIssuerKeyGen(pp) \rightarrow (VCISK, VCIPK)$: Executed by the VC issuer, input the public parameters pp, and output the VC issuer's private key $VCISK$ and public key $VCIPK$.

(3) **VC Request Generation** $VCRequestGen(pp, m, VCIPK, VCDIDs) \rightarrow (VCReq)$: Executed by the user, input the public parameters pp, attribute information m, the VC issuer's public key $VCIPK$, and the identifier $VCDIDs$ used when applying for the VC. Output the VC request $VCReq$.

(4) **VC Generation** $VCGen(pp, VCReq, VCISK, VCIPK) \rightarrow (VC)$: Executed by the VC issuer, input the public parameters pp, the user's VC request $VCReq$, the VC issuer's private key $VCISK$, and the VC issuer's public key $VCIPK$. Output VC signed by the VC issuer.

(5) **VC Revocation** $VCRev(VCISK, VCIPK, VC) \rightarrow (VCIPK')$: Executed by the VC issuer, input the VC issuer's private key $VCISK$, the VC issuer's public key $VCIPK$, and the VC to be revoked. Output the updated VC issuer's public key $VCIPK'$.

(6) **VC Update** $VCUpdate(VCIPK', VC) \rightarrow (VC')$: Executed by the user with a non-revoked VC, input the updated VC issuer's public key $VCIPK'$ and their VC. Output the updated VC', which the user needs to use for future presentations.

(7) **VP Generation** $VPGen(pp, VCIPKSet, VCSet, dSet, VPDIDs, d_{vp}) \rightarrow (VP)$: Executed by the user, input the public parameters pp, the set of VCs to be combined into the VP $VCSet$, the set of public keys corresponding to the VCs $VCIPKSet$, the set of keys corresponding to the VCs' DIDs $dSet$,

the identifier $VCDIDs$ used for the VP presentation, and the key d_{vp} corresponding to this identifier. Output the selectively disclosed verifiable presentation VP.

(8) **VP Verification** $VPVer(pp, VCIPKSet, VP, VPDIDs) \rightarrow (0/1)$: Executed by the verifier, input the public parameters pp, the selectively disclosed presentation VP, the set of public keys of the VC issuers included in the presentation $VCIPKSet$, and the identifier $VCDIDs$ used for the presentation. Output 0 or 1, where 1 indicates the verification is successful, and 0 indicates failure.

6.2 Security model

The security properties of the SR-PP-VC scheme include anonymity, unforgeability, regulatory reliability, and revocation reliability. This scheme constructs a formal definition of security properties using a single VC issuer and a VP composed of a single VC. If an extension to multiple VC issuers and VPs composed of multiple VCs is required, the corresponding public and private keys for the issuers need to be generated. The scheme uses the following lists and oracles to describe the security model:

$\mathcal{L}_{H\_U}, \mathcal{L}_{C\_U}$: Lists of honest and corrupt users, respectively, recording the indices i of honest and corrupt users, $\mathcal{L}_{H\_U} \cap \mathcal{L}_{C\_U} = \varnothing$.

$\mathcal{L}_{DIDs}, \mathcal{L}_{DIDs\_sk}$: Lists of users' DIDs (including MDIDs and PDIDs) and the corresponding keys.

$\mathcal{L}_{Owner}, \mathcal{L}_{VC}, \mathcal{L}_m, \mathcal{L}_{VCDIDs}$: Lists of VC owners, VCs, VC attribute sets, and DIDs used to apply for VCs, respectively. Elements in $\mathcal{L}_{Owner}, \mathcal{L}_{VC}, \mathcal{L}_m$ *and* $\mathcal{L}_{VCDIDs}$ correspond one-to-one.

$\mathcal{L}_{VP}, \mathcal{L}_{VPDIDs}$: Lists of VPs and the DIDs used to present the VPs, recording the VPs that users have presented and the DIDs used during the presentation.

$\mathcal{L}_{RevVC}$: List of revoked VCs.

Oracle for Honest Users $\mathcal{O}_{H\_U}(i)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i to add new user i, if $i \notin \mathcal{L}_{H\_U} \cup \mathcal{L}_{C\_U}$, executing $\mathcal{L}_{H\_U} = \mathcal{L}_{H\_U} \cup \{i\}$.

Oracle for Corrupt Users $\mathcal{O}_{C\_U}(i)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i to obtain all identity information of the user, including the VCs the user owns, the set of DIDs, and the corresponding keys.

Oracle for VC $\mathcal{O}_{VC}(i, m)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i and attribute information m to generate a VC for the user. If $i \notin \mathcal{L}_{H\_U}$, return $\perp$. Otherwise, randomly select $VCDIDs \in \mathcal{L}_{DIDs}[i]$ and ensure $VCDIDs \cap \mathcal{L}_{VCDIDs} = \varnothing$. Execute the $VCRequestGen$ and $VCGen$ algorithms, updating $(\mathcal{L}_{Owner}, \mathcal{L}_{VC}, \mathcal{L}_m, \mathcal{L}_{VCDIDs}) = (\{i\}, \{VC\}, \{m\}, \{VCDIDs\}) \cup (\mathcal{L}_{Owner}, \mathcal{L}_{VC}, \mathcal{L}_m, \mathcal{L}_{VCDIDs})$, and return $\top$.

Oracle for VC Request $\mathcal{O}_{VCReq}(i, m)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i and attribute information m to obtain a VC request. If $i \notin \mathcal{L}_{H\_U}$, return $\perp$. Otherwise, randomly select $VCDIDs \in \mathcal{L}_{DIDs}[i]$ and ensure $VCDIDs \cap \mathcal{L}_{VCDIDs} = \varnothing$. Execute the $VCRequestGen$ algorithm, updating $(\mathcal{L}_{Owner}, \mathcal{L}_{VC}, \mathcal{L}_m, \mathcal{L}_{VCDIDs}) = (\{i\}, \{\perp\}, \{m\}, \{VCDIDs\}) \cup (\mathcal{L}_{Owner}, \mathcal{L}_{VC}, \mathcal{L}_m, \mathcal{L}_{VCDIDs})$, and return $VCReq, VCDIDs$.

Oracle for VC Generation $\mathcal{O}_{VCGen}(i, m, VCDIDs, VCReq)$: Adversary $\mathcal{A}$ queries this oracle with user identifier i, attribute information m, the identifier $VCDIDs$ used to apply for the VC, and the VC request parameters $VCReq$ to obtain a VC. If $i \notin \mathcal{L}_{H\_U}$, return $\perp$. Otherwise, execute the $VCGen$ algorithm, updating $(\mathcal{L}_{Owner}, \mathcal{L}_{VC}, \mathcal{L}_m, \mathcal{L}_{VCDIDs}) = (\{i\}, \{VC\}, \{m\}, \{VCDIDs\}) \cup (\mathcal{L}_{Owner}, \mathcal{L}_{VC}, \mathcal{L}_m, \mathcal{L}_{VCDIDs})$, and return VC.

Oracle for VP Generation $\mathcal{O}_{VP}(j)$: Adversary $\mathcal{A}$ queries this oracle with VC index j to obtain a VP. Assume $i = \mathcal{L}_{Owner}[j]$. If $i \notin \mathcal{L}_{H\_U}$ or $\mathcal{L}_{VC}[j] \in \mathcal{L}_{RevVC}$, return $\perp$. Otherwise, set $VCDIDs = \mathcal{L}_{VCDIDs}[j]$ and d_{vc} as the corresponding key. Randomly select $VPDIDs \in \mathcal{L}_{DIDs}[i]$ and its corresponding key d_{vp}, ensuring $VPDIDs \cap \mathcal{L}_{VPDIDs} = \varnothing$. Execute $VPGen(pp, VCIPK, \mathcal{L}_{VC}[j], d_{vc}, VPDIDs, d_{vp}) \rightarrow (VP)$, and update $(\mathcal{L}_{VP}, \mathcal{L}_{VPDIDs}) = (\mathcal{L}_{VP}, \mathcal{L}_{VPDIDs}) \cup (\{VP\}, \{VPDIDs\})$, then return VP, $VPDIDs$.

Oracle for VC Revocation $\mathcal{O}_{VCRev}(j)$: Adversary $\mathcal{A}$ queries this oracle with VC index j to revoke the VC. Execute $VCRev(VCISK, VCIPK, \mathcal{L}_{VC}[j]) \rightarrow (VCIPK)$ and update $\mathcal{L}_{RevVC} = \mathcal{L}_{RevVC} \cup \{\mathcal{L}_{VC}[j]\}$. For all $k < \mathcal{L}_{VC}.length$ and $k \neq j$, execute $(VCUpdate(VCIPK, \mathcal{L}_{VC}[k]) \rightarrow \mathcal{L}_{VC}[k])$. Finally, return $VCIPK$.

(1) Anonymity

In the SR-PP-VC scheme, anonymity is defined as except for the regulator, no other entities can infer a user's identity information through DIDs. Additionally, VC issuers and verifiers can only access

the necessary disclosed information within the user's VC or VP and cannot learn more about the user's identity attributes. Even if the same credential is used with different DIDs to disclose different attributes at different times, verifiers cannot link these presentations to the same user or credential, ensuring the unlinkability of DIDs and VPs.

Definition 4. For any probabilistic polynomial-time adversary $\mathcal{A}$, if $\mathcal{A}$ can make a polynomially bounded number of queries to the oracle list $\mathcal{O} = \{\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{VC}, \mathcal{O}_{VCReq}, \mathcal{O}_{VP}\}$, and the probability in Equation (4) is negligible, then the SR-PP-VC scheme satisfies anonymity.

$$Pr\left[\begin{array}{c} SetUp\left(1^{\lambda}\right) \rightarrow pp; \\ VCIssuerKeyGen(pp) \rightarrow (VCISK, VCIPK); \\ \mathcal{A}^{\mathcal{O}}(pp, VCIPK) \rightarrow \{i_0, i_1, m^*\}; \\ \mathcal{C}\left(pp, i_b, m^*\right) \rightarrow \left(VP_{i_b}, VPDIDs_{i_b}\right); \\ \mathcal{A}\left(pp, VP_{i_b}, VPDIDs_{i_b}\right) \rightarrow b^* \end{array} : b^* = b\right] - \frac{1}{2} \tag{4}$$

(2) Unforgeability

In the SR-PP-VC scheme, unforgeability is defined as without the VC issuer's private key, no attacker can generate a verifiable VP, even if they have obtained multiple VPs from honest users. Additionally, users cannot modify any attribute values in an issued VC.

Definition 5. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can make a polynomially bounded number of queries to the oracle list $\mathcal{O} = \{\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{VC}, \mathcal{O}_{VCGen}, \mathcal{O}_{VP}\}$, and the probability in Equation (5) is negligible, then the SR-PP-VC scheme satisfies unforgeability.

$$Pr\left[\begin{array}{c} SetUp\left(1^{\lambda}\right) \rightarrow pp; \\ VCIssuerKeyGen(pp) \rightarrow (VCISK, VCIPK); \\ \mathcal{A}^{\mathcal{O}}(pp, VCIPK) \rightarrow (VP^*, VPDIDs^*, m^*); \\ VPVer\left(pp, VCIPK, VP^*, VPDIDs^*\right) \rightarrow b^* : \\ (b^* = 1) \wedge (m^* \notin \mathcal{L}_m \vee \\ (VP^* \notin \mathcal{L}_{VP} \wedge VPDIDs^* \notin \mathcal{L}_{VPDIDs})) \end{array}\right] \tag{5}$$

(3) Regulatory Reliability

In the SR-PP-VC scheme, regulatory reliability is defined as the regulator can correctly trace a user's real identity information from the DIDs associated with the selectively disclosed VC.

(4) Revocation Reliability

In the SR-PP-VC scheme, revocation reliability is defined as: users with revoked VCs cannot forge a valid VC.

Definition 6. For any PPT adversary $\mathcal{A}$, if $\mathcal{A}$ can make a polynomially bounded number of queries to the oracle list $\mathcal{O} = \{\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{VC}, \mathcal{O}_{VCGen}, \mathcal{O}_{VP}, \mathcal{O}_{VCRev}\}$, and the probability in Equation (6) is negligible, then the SR-PP-VC scheme satisfies revocation reliability.

$$Pr\left[\begin{array}{c} SetUp\left(1^{\lambda}\right) \rightarrow pp; \\ VCIssuerKeyGen(pp) \rightarrow (VCISK, VCIPK); \\ \mathcal{A}^{\mathcal{O}}\left(pp, VCIPK, VC \in \mathcal{L}_{RevVC}, VPDIDs\right) \rightarrow (VP^*); \\ VPVer\left(pp, VCIPK, VP^*, VPDIDs\right) \rightarrow b^* : \\ b^* = 1 \end{array}\right] \tag{6}$$

6.3 Scheme design

(1) $SetUp(1^{\lambda}) \rightarrow pp$: This algorithm is the same as the $SetUp$ algorithm in the SR-DIDs scheme. Output public parameters $pp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbb{Z}_p, g_1, g_2, e, \mathcal{H}, DR)$.

(2) $VCIssuerKeyGen(pp) \rightarrow (VCISK, VCIPK)$: The VC issuer selects an integer $x \in \mathbb{Z}_p$ and computes $\overline{X} = g_2^x$. Assume the number of attributes in the issued VC is l, set the attribute name list as $AttrNames = [name_1, \ldots, name_l]$, and randomly choose $HATTR = (HAttr_1, \ldots, HAttr_l) \leftarrow \mathbb{G}_1$. Then choose $\widetilde{g}_1 \leftarrow \mathbb{G}_1$, initialize the accumulator $\Delta = \widetilde{g}_1$, and generate an empty VC revocation list

$VCRList = \varnothing$. Finally, output the VC issuer's private key $VCISK = x$ and public key $VCIPK = (\overline{X}, AttrNames, HATTR, \Delta, VCRList)$. The VC issuer stores $VCISK$ locally and uploads $VCIPK$ to the blockchain in the issuer's corresponding DID document.

(3) $VCRequestGen(pp, m, VCIPK, VCDIDs) \rightarrow (VCReq)$: Users can apply for a VC using any of their DIDs. Assume the DID used is $VCDIDs = DSK^{umsk} \cdot DRand^{d_{vc}}$. The user provides their attribute information m based on the attribute name list $AttrName$, let $m = (m_1, \ldots, m_l) \in \mathbb{Z}_p$. The user's attribute values are divided into two types: attributes to be hidden from the VC issuer (set HI) and attributes that do not need to be hidden (set NHI). The user computes the commitment of the hidden attributes: $C_{HI} = g_1 \prod_{i \in HI} HAttr_i{}^{m_i}$. The VC request parameters are $VCReq = (C_{HI}, (m_i)_{i \in NHI})$, which are sent to the VC issuer via $VCDIDs$.

(4) $VCGen(pp, VCReq, VCISK, VCIPK) \rightarrow (VC)$:

① The VC issuer verifies the validity of the applicant's $VCDIDs$ and $(m_i)_{i \in NHI}$. If valid, the issuer signs the user's attributes. First, compute $C = C_{HI} \cdot VCDIDs \cdot \Delta \cdot \prod_{i \in NHI} HAttr_i{}^{m_i}$. Then, select an integer $k \in \mathbb{Z}_p$ and compute the BBS signature $A = C^{1/(x+k)}$ and the validity proof $Valid = \Delta^{1/(x+k)}$. Finally, send the attribute signature $Sign = (A, C, k, Valid)$ to the user and store $(Sign, VCDID, k, C_{HI}, (m_i)_{i \in NHI})$ in the local database.

② Upon receiving $Sign$, the user can verify its validity by checking if $C = g_1 \cdot VCDIDs \cdot \Delta \cdot \prod_i^l HAttr_i{}^{m_i}$ and $e(A \cdot Valid, \overline{X}) = e(C \cdot A^{-k} \cdot \Delta \cdot Valid^{-k}, g_2)$ hold. If valid, the user generates the VC certificate $VC = (A, C, k, Valid, m)$ and stores it locally; otherwise, the signature is invalid.

(5) $VCRev(VCISK, VCIPK, VC) \rightarrow (VCIPK')$: The VC issuer first finds the information related to the VC to be revoked $(Sign', VCDIDs', k', C_{HI}{}', (m_i')_{i \in NHI})$ in the local database. Set $VCRList = VCRList \cup \{k'\}$ and update the revocation accumulator $\Delta_{new} = \Delta^{1/(x+k')}$. Finally, update the issuer's public key on the blockchain to $VCIPK' = (\overline{X}, AttrNames, HATTR, \Delta_{new}, VCRList)$.

(6) $VCUpdate(VCIPK', VC) \rightarrow (VC')$: Users can check if the issuer's public key has been updated. If updated and the revoked VCs does not belong to them, they need to update their VC. First, update the validity proof $Valid_{new} = (Valid/\Delta_{new})^{1/(k'-k)}$. Then, update the VC signature $A_{new} = A \cdot Valid_{new}/Valid$. Finally, update their VC to $VC' = (A_{new}, k, Valid_{new}, m)$.

(7) $VPGen(pp, VCIPKSet, VCSet, dSet, VPDIDs, d_{vp}) \rightarrow (VP)$: Users can combine n VCs into a single VP, using any DID for presentation, and selectively disclose attributes. The variables in this algorithm are explained in Table 1.

① Compute $C_{DI_j} = g_1 \cdot VPDIDs \cdot \Delta_j \cdot \prod_{i \in DI_j} HAttr_{j,i}{}^{m_{j,i}}$ for all $j \in N$. Randomly select $r_3 \in \mathbb{Z}_p$ and compute $\overline{A}_j = A_j^{r_3}$ for all $j \in N$, $\overline{F}_j = (C_j \cdot A_j^{-k_j})^{r_3}$ for all $j \in N$, and let $\overline{G} = \prod_{j \in N} \overline{F}_j$, $t_{dj} = d_{vcj} - d_{vp}$ for all $j \in N$.

② Compute the zero-knowledge proof $\pi_2 = NIZK - PoK_2\{r_3, (k_j)_{j \in N}, (t_{dj})_{j \in N}, (m_{j,i})_{j \in N, i \in NDI_j} | \overline{G} = \prod_{j \in N}((C_{DI_j} \cdot DRand^{t_{dj}} \cdot \prod_{i \in NDI} HAttr_{j,i}{}^{m_{j,i}})^{r_3} \cdot \overline{A}_j{}^{-k_j})\}$: Randomly select $\overline{r}_3$, $(\overline{r}_{k_j})_{j \in N}$, $(\overline{r}_{t_j})_{j \in N}$, $(\overline{\delta}_{j,i})_{j \in N, i \in NDI_j} \in \mathbb{Z}_p$, and compute the auxiliary value $\overline{G}' = \prod_{j \in N} (C_{DI_j}{}^{\overline{r}_3} \cdot \overline{A}_j^{-\overline{r}_{k_j}} \cdot DRand^{\overline{r}_{t_j}} \cdot \prod_{i \in NDI_j} HAttr_{j,i}{}^{\overline{\delta}_{j,i}})$. Compute the challenge value $c_2 = \mathcal{H}(VPDIDs, (\overline{A}_j)_{j \in N}, \overline{G}, \overline{G}')$ and $s_{r_3} = \overline{r}_3 + c_2 \cdot r_3$, $s_{k_j} = \overline{r}_{k_j} + c_2 \cdot k_j$, $s_{t_j} = \overline{r}_{t_j} + c_2 \cdot t_{dj} \cdot r_3$ and $s_{\delta_{j,i}} = \overline{\delta}_{j,i} + c_2 \cdot r_3 \cdot m_{j,i}$ for all $j \in N$ and $i \in NDI_j$. Let $\pi_2 = (c_2, s_{r_3}, (s_{k_j})_{j \in N}, (s_{t_j})_{j \in N}, (s_{\delta_{j,i}})_{j \in N, i \in NDI_j})$.

③ Finally, generate $VP = ((\overline{A}_j)_{j \in N}, \overline{G}, (m_{j,i})_{j \in N, i \in DI_j}, \pi_2)$ and send VP to the verifier via $VPDIDs$.

(8) $VPVer(pp, VCIPKSet, VP, VPDIDs) \rightarrow (0/1)$: Upon receiving the VP, the verifier first checks if $\prod_{j \in N}(e(\overline{A}_j, \overline{X}_j)) = e(\overline{G}, g_2)$. If not, the VP is invalid. Otherwise, proceed to zero-knowledge proof verification:

① Compute the auxiliary value $C_{DI_j} = g_1 \cdot VPDIDs \cdot \Delta_j \cdot \prod_{i \in DI_j} HAttr_{j,i}{}^{m_{j,i}}$ for all $j \in N$ and $\overline{G}'' = \overline{G}^{-c_2} \cdot \prod_{j \in N} (C_{DI_j}{}^{s_{r_3}} \cdot \overline{A}_j^{-s_{k_j}} \cdot DRand^{s_{t_j}} \cdot \prod_{i \in NDI_j} HAttr_{j,i}{}^{s_{\delta_{j,i}}})$, where Δ_j should be the latest value on the blockchain.

② Compute the challenge value $c_2' = \mathcal{H}(VPDIDs, (\overline{A}_j)_{j \in N}, \overline{G}, \overline{G}'')$;

③ Verify if $c_2' = c_2$. If so, the VP is valid, output 1; otherwise, output 0.

Table 1. Variable definition and explanation

Variable	Explanation
N	$N = \{1, 2, \ldots, n\}$
$VCSet$	Set of n VCs to be combined into a VP, $VCSet = \{VC_j\}$ for all $j \in N$, where $VC_j = (A_j, C_j, k_j, Valid_j, m_j)$
$VCIPKSet$	Set of VC issuer public keys, $VCIPKSet = \{VCIPK_j\}$ for all $j \in N$ where $VCIPK_j = (\bar{X}_j, AttrNames_j, HATTR_j, \Delta_j, VCRList_j)$
$dSet$	Set of DID keys, $dSet = \{d_{vcj}\}$ for all $j \in N$. Let $VCDIDs_j$ be the DIDs used when applying for the j-th VC, then $VCDIDs_j = DSK^{msk} \cdot DRand^{d_{vcj}}$
$m_{j,i}$	The i-th attribute of the j-th VC
DI_j	The set of attribute indices to be disclosed for the j-th VC
NDI_j	The set of attribute indices not to be disclosed for the j-th VC
NDI	$NDI = \{NDI_j\}$ for all $j \in N$
$VPDIDs$	The DID used for presenting the VP, $VPDIDs = DSK^{msk} \cdot DRand^{d_{vp}}$

6.4 Security analysis

(1) Anonymity

Theorem 4. If the SR-DIDs scheme satisfies anonymity, the Pedersen commitment scheme satisfies hiding, and the Fiat-Shamir non-interactive zero-knowledge (NIZK) proof protocol satisfies zero-knowledge, then the SR-PP-VC scheme satisfies anonymity.

Proof. If an adversary $\mathcal{A}$ can successfully attack the anonymity of the SR-PP-VC scheme with non-negligible probability, then it is possible to construct an algorithm $\mathcal{B}$ that can attack the anonymity of the SR-DIDs scheme, the hiding property of the Pedersen commitment, or the zero-knowledge property of the Fiat-Shamir NIZK proof protocol with the same probability. The interaction between $\mathcal{A}$ and $\mathcal{B}$ is as follows:

① **Setup Phase**: $\mathcal{B}$ constructs the system parameters pp and the VC issuer's public key $VCIPK$ for the SR-PP-VC scheme based on the anonymity of the SR-DIDs scheme, the hiding property of the Pedersen commitment, and the zero-knowledge property of the NIZK proof protocol. $\mathcal{B}$ then sends $(pp, VCIPK)$ to $\mathcal{A}$.

② **Query Phase I**: $\mathcal{A}$ can make a polynomially bounded number of queries to $\mathcal{B}$. $\mathcal{B}$ follows the procedures of the oracles $\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{VC}, \mathcal{O}_{VCReq}, \mathcal{O}_{VP}$ and returns the results to $\mathcal{A}$. During the generation of DIDs, the calculation of hidden attribute commitments, and the generation of zero-knowledge proofs, $\mathcal{B}$ obtains results from the anonymity of the SR-DIDs scheme, the Pedersen commitment scheme, and the NIZK proof protocol simulation games.

③ **Challenge Phase**: $\mathcal{A}$ first selects two honest users i_0, i_1 and attribute information m^* and sends them to $\mathcal{B}$. $\mathcal{B}$ then forwards them to the SR-DIDs anonymity, Pedersen commitment hiding, and NIZK proof protocol simulation games, obtaining the results $(SVC_{i_b}, SVCDIDs_{i_b})$, and sends them to $\mathcal{A}$.

④ **Query Phase II**: $\mathcal{A}$ continues querying as in Query Phase I but cannot query the oracle $\mathcal{O}_{C\_U}$ for users i_0, i_1.

⑤ **Guessing Phase**: $\mathcal{A}$ guesses b and outputs b'. $\mathcal{B}$ forwards b' to the SR-DIDs anonymity, Pedersen commitment hiding, and NIZK-proof can successfully attack the anonymity of the SR-DIDs scheme, the hiding property of the Pedersen commitment, or the zero-knowledge property of the Fiat-Shamir NIZK proof protocol, then $\mathcal{A}$ can succeed in the anonymity experiment of the SR-PP-VC scheme. However, since the SR-DIDs scheme satisfies anonymity, the Pedersen commitment scheme satisfies hiding, and the Fiat-Shamir NIZK proof protocol satisfies zero-knowledge, the probability of $\mathcal{A}$ winning is negligible, *i.e.*, the probability in Equation (4) is negligible. Therefore, the SR-PP-VC scheme satisfies anonymity.

(2) Unforgeability

Theorem 5. If the BBS signature scheme satisfies SUF-CMA security, the Pedersen commitment scheme satisfies binding, and the zero-knowledge proof π_2 satisfies zero-knowledge, completeness, and soundness, then the SR-PP-VC scheme satisfies unforgeability.

Proof. Suppose $\mathcal{A}$ is an adversary of the unforgeability of the SR-PP-VC scheme, $\mathcal{B}$ is an algorithm that attacks the SUF-CMA security of the BBS signature scheme, the binding property of the Pedersen commitment, or the zero-knowledge, completeness, and soundness of the Fiat-Shamir NIZK proof protocol, and $\mathcal{C}_{BBS}$ is the challenger of the SUF-CMA security of the BBS signature scheme, $\mathcal{C}_{Pedersen}$ is the

challenger of the binding property of the Pedersen commitment, and $\mathcal{S}$ is the simulator for generating the zero-knowledge proof π_2. The interaction between $\mathcal{A}$ and $\mathcal{B}$ is as follows:

① **Setup Phase**: $\mathcal{C}_{BBS}$ generates the public parameters $(g_1, g_2, \overline{X} = g_2^x)$ for the SUF-CMA security game of the BBS signature scheme, and $\mathcal{C}_{Pedersen}$ generates the public parameters $(HAttr_1, \ldots, HAttr_l)$ for the binding property game of the Pedersen commitment. $\mathcal{B}$ constructs $(pp, VCIPK)$ based on these public parameters and sends them to $\mathcal{A}$.

② **Query Phase**: $\mathcal{A}$ can make q queries to $\mathcal{B}$. $\mathcal{B}$ follows the procedures of the oracles $\mathcal{O}_{H\_U}, \mathcal{O}_{C\_U}, \mathcal{O}_{VC}, \mathcal{O}_{VCGen}, \mathcal{O}_{VP}$. For parts involving signatures, $\mathcal{B}$ constructs the commitment value to be signed and sends it to $\mathcal{C}_{BBS}$ for signing, then sends the result to $\mathcal{B}$. For parts involving attribute commitment calculations, $\mathcal{C}_{Pedersen}$ performs the calculations and sends the results to $\mathcal{B}$. For parts involving the generation of the zero-knowledge proof π_2, $\mathcal{S}$ runs the simulation. Finally, $\mathcal{B}$ returns the results to $\mathcal{A}$.

③ **Forgery Phase**: $\mathcal{A}$ constructs a verifiable presentation VP^* about the attribute information m^* and its corresponding identifier $VPDIDs^*$ for user i, and sends $(VP^*, VPDIDs^*, m^*)$ to $\mathcal{B}$.

④ **Judgment Phase**: If VP^* passes verification and $(m^* \notin \mathcal{L}_m \vee (VP^* \notin \mathcal{L}_{VP} \wedge VPDIDs^* \notin \mathcal{L}_{VPDIDs}))$, then $\mathcal{A}$ has successfully attacked. At this point, $\mathcal{B}$ forwards $(\overline{A}^*, \overline{G}^*)$ to $\mathcal{C}_{BBS}$ and $\mathcal{C}_{Pedersen}$, and the knowledge proof π_2^* to $\mathcal{S}$.

It is evident that $\mathcal{A}$'s observations in the SR-PP-VC scheme's unforgeability experiment are equivalent to those in $\mathcal{B}$'s experiment. Since both $\mathcal{A}$ and $\mathcal{B}$ are PPT algorithms, the probability of $\mathcal{A}$ successfully attacking the unforgeability of the SR-PP-VC scheme is the same as the probability of $\mathcal{B}$ successfully attacking the SUF-CMA security of the BBS signature scheme, the binding property of the Pedersen commitment, or the zero-knowledge, completeness, and soundness of the Fiat-Shamir NIZK proof protocol. However, since the BBS signature scheme satisfies SUF-CMA security, the Pedersen commitment satisfies binding, and the Fiat-Shamir NIZK proof protocol satisfies zero-knowledge, completeness, and soundness, the probability in Equation (5) is negligible. Therefore, the SR-PP-VC scheme satisfies unforgeability.

(3)Regulatory Reliability

Theorem 6. If the SR-DIDs scheme satisfies regulatory reliability, then the SR-PP-VC scheme satisfies regulatory reliability.

Proof. Since the DIDs in the SR-PP-VC scheme are generated based on the SR-DIDs scheme, and users must disclose the corresponding DIDs each time they apply for and present a credential, the regulator can reliably monitor user identity information.

(4) Revocation Reliability

Theorem 7. If the BBS signature scheme satisfies SUF-CMA security and the zero-knowledge proof π_2 satisfies zero-knowledge, completeness, and soundness, then the SR-PP-VC scheme satisfies revocation reliability.

Proof. For the specific scheme, when a user attempts to construct a false VP with selective disclosure, there are three possible cases:

① **Case 1**: The user continues to generate the VP as usual. Since $\Delta_{new} \neq \Delta$, the user's calculated $C_{DI} = g_1 \cdot VPDIDs \cdot \Delta \cdot \prod_{i \in DI} HAttr_i{}^{m_i}$ will not equal the verifier's calculated $C_{DI} = g_1 \cdot VPDIDs \cdot \Delta_{new} \cdot \prod_{i \in DI} HAttr_i{}^{m_i}$. Consequently, $\overline{G} \neq \overline{G}'$, and due to the collision resistance of the hash function, $c_2 \neq c_2'$.

② **Case 2**: The user calculates $C_{DI} = g_1 \cdot VPDIDs \cdot \Delta_{new} \cdot \prod_{i \in DI} HAttr_i{}^{m_i}$ using Δ_{new}. However, $A \neq (C_{DI} \cdot DRand^{t_3} \cdot \prod_{i \in NDI} HAttr_i{}^{m_i})^{1/(x+k)}$. Therefore, $\overline{G} \neq \overline{G}'$, and $c_2 \neq c_2'$.

③ **Case 3**: The user can construct a VP based on the revoked VC that passes verification. This implies the ability to break the SUF-CMA security of the BBS signature scheme. In this scheme, the accumulator acts as one of the attributes signed by the VC issuer. If verification is successful, it indicates that a forged signature $m^* \neq m_i$ was created in the SUF-CMA security game of the BBS signature scheme, thus breaking its SUF-CMA security.

In conclusion, the SR-PP-VC scheme satisfies revocation reliability.

Table 2. Comparison of security between SR-DIDs and related schemes

	Anonymity	Unforgeability	Unlinkability	Regulatory	Regulatory reliability	Revocation reliability
IDDKSAP [7]	√	√	√	×	×	×
ProChain [8]	√	√	√	×	×	×
The scheme in reference [9]	√	√	×	√	√	×
The scheme in reference [10]	√	×	√	√	×	×
SR-DIDs	√	√	√	√	√	√

Table 3. Comparison of security between SR-PP-VC and related schemes

Scheme	Anonymity	Unforgeability	Unlinkability	Regulatory	Regulatory reliability	Revocation reliability	Selective disclosure	Constant-size signatures
The scheme in reference [12]	√	√	√	×	×	×	×	√
The scheme in reference [13]	√	√	√	√	×	×	√	√
tsrCert [15]	√	√	×	√	×	×	√	×
BASS [16]	√	√	√	×	×	√	×	√
Protego [17]	√	√	√	×	×	√	×	√
CredChain [22]	√	√	√	×	×	×	√	×
CanDID [23]	√	√	√	×	×	×	×	√
The scheme in reference [24]	√	√	√	×	×	×	√	×
Idemix	√	√	√	×	×	×	√	√
SR-PP-VC	√	√	√	√	√	√	√	√

7 Scheme analysis and comparison

7.1 Security comparison

The security comparison of the SR-DIDs scheme, SR-PP-VC scheme, and related works are shown in Tables 2 and 3, respectively. It can be seen that, compared to other similar schemes, the proposed schemes in this paper provide more comprehensive security and privacy protection measures.

7.2 Performance testing

The experiments were conducted on a PC running Windows 10 Professional 64-bit, equipped with Intel(R) Core(TM) i7-11700 @ 2.50GHz processor and 16GB of RAM. The simulation tests used Hyperledger Fabric blockchain version 2.5.4, deployed via Docker with 2 peer nodes and 1 order node. The development was done in JAVA, utilizing the BLS12-381 bilinear pairing curves [36], implemented using the MIRACL library, and the SHA-384 algorithm for the hash function $\mathcal{H}$.

(1) Performance comparison of SR-DIDs scheme

This study implemented the ring signature-based regulatory scheme proposed in Reference [9] and the BBS04-SR-DIDs scheme designed based on the BBS04 group signature [14], as benchmarks for evaluating the performance of the SR-DIDs scheme. The BBS04-SR-DIDs scheme uses the original BBS04 group signature to construct the generation and verification mechanism of PDIDs, while SR-DIDs employs an optimized group signature algorithm, demonstrating the effect of optimization.

For example, the space complexity comparison of generating a single proxy identity identifier among the three schemes is shown in Table 4, where $\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ represent the element lengths in the groups $\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively. In the BLS12-381 curve, their sizes are 48 bytes, 48 bytes, 96 bytes, and 576 bytes, respectively, and k is the number of ring signature members. In the BLS12-381 curve, the SR-DIDs scheme shows an increase in the space complexity of the regulator's key pair and the user's key compared to Reference [9], but since it is generated once and can be reused, the additional 336 bytes of space is negligible compared to modern storage capacities. In Reference [9], the complexity of identity generation increases with the number of ring signature members k, and its security is positively correlated with k, leading to higher overall space consumption. The SR-DIDs scheme and the BBS04-SR-DIDs scheme have limited differences, with only an additional 48 bytes.

The time complexity of identity generation, verification, and regulation for the three schemes is also compared, as shown in Table 5, and implemented and tested through code. Here, E_1, M_1 represent

Table 4. Comparison of space complexity between SR-DIDs and related schemes

	SR-DIDs	BBS04-SR-DIDs	The scheme in reference [9]
Supervisor key pair	$3\,\lvert\mathbb{Z}_p\rvert+3\,\lvert\mathbb{G}_1\rvert+1\lvert\mathbb{G}_2\rvert$	$3\,\lvert\mathbb{Z}_p\rvert+3\,\lvert\mathbb{G}_1\rvert+1\lvert\mathbb{G}_2\rvert$	$1\,\lvert\mathbb{Z}_p\rvert+1\,\lvert\mathbb{G}_1\rvert+1\lvert\mathbb{G}_2\rvert$
User key	$3\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_1\rvert$	$3\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_1\rvert$	$1\lvert\mathbb{Z}_p\rvert$
Identity identifier	$1\lvert\mathbb{G}_1\rvert$	$1\lvert\mathbb{G}_1\rvert$	$1\lvert\mathbb{G}_1\rvert$
Identity generation	$5\lvert\mathbb{Z}_p\rvert+5\lvert\mathbb{G}_1\rvert$	$6\lvert\mathbb{Z}_p\rvert+3\lvert\mathbb{G}_1\rvert$	$2k\lvert\mathbb{Z}_p\rvert+1\lvert\mathbb{G}_T\rvert$

Table 5. Comparison of time complexity between SR-DIDs and related schemes

	SR-DIDs	BBS04-SR-DIDs	The scheme in reference [9]
Identity generation	$14E_1+7M_1+1H$	$10E_1+4M_1+3E_T+2M_T+3P+1H$	$(2k-1)\,E_1+(k-1)\,M_1+2kE_T+(k-1)\,M_T+(k+1)P+1H$
Identity authentication	$11E_1+8M_1+2P+1H$	$8E_1+4M_1+4E_T+4M_T+5P+1H$	$2kE_1+kM_1+2kE_T+kM_T+kP+1H$
Identity regulation	$2E_1+2M_1$	$2E_1+2M_1$	$(1E_T+1P)\sim(kE_T+kP)$

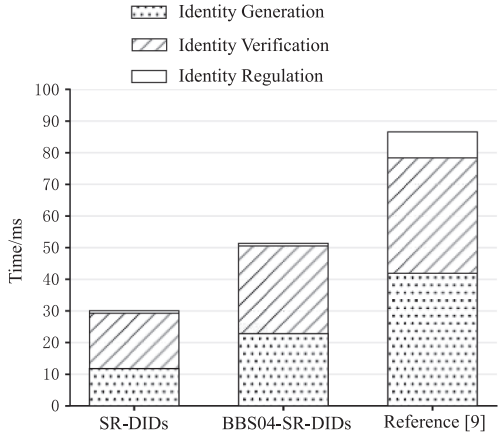


Figure 3. Time comparison of identity identifier management

Table 6. Comparison of space complexity between SR-PP-VC and idemix

	SR-PP-VC	Idemix
VC issuer key pair	$\mathbf{1}\,\lvert\mathbb{Z}_p\rvert+(l+1)\,\lvert\mathbb{G}_1\rvert+1\lvert\mathbb{G}_2\rvert$	$\mathbf{3}\,\lvert\mathbb{Z}_p\rvert+(l+4)\,\lvert\mathbb{G}_1\rvert+1\lvert\,\mathbb{G}_2\rvert$
VC store	$\mathbf{1}\,\lvert\mathbb{Z}_p\rvert+3\lvert\,\mathbb{G}_1\rvert$	$\mathbf{2}\,\lvert\mathbb{Z}_p\rvert+2\lvert\,\mathbb{G}_1\rvert$
VP generation	$(2n+2+h)\,\lvert\mathbb{Z}_p\rvert+(n+1)\,\lvert\mathbb{G}_1\rvert$	$(h+8n)\,\lvert\mathbb{Z}_p\rvert+4n\lvert\mathbb{G}_1\rvert$

exponentiation and multiplication operations in the group $\mathbb{G}_1$, E_T, M_T represent exponentiation and multiplication operations in the group $\mathbb{G}_T$, P represents bilinear pairing operations, and H represents hash operations. The experimental design follows the principle that the number of ring signature members in practical applications usually does not exceed 10, with $k = 5$ chosen for testing. Figure 3 shows the average time results for 100 executions of each scheme's algorithm. The identity generation, verification, and regulation times for the SR-DIDs scheme are approximately 11.8 ms, 17.5 ms, and 0.81 ms, respectively.

(2) Performance comparison of SR-PP-VC scheme

This study compares the space complexity of the SR-PP-VC scheme with the anonymous certificate management scheme (Idemix) on Hyperledger Fabric, using examples of issuer key pair storage, single VC storage, and VP generation, as shown in Table 6. The parameters l represent the number of attributes in a single credential, n represents the number of VCs in the VP, and h represents the total number of hidden attributes in the VP. Although Idemix does not have a direct implementation of VP, VP is essentially multiple VCs combined, so the complexity of managing a single credential in Idemix multiplied by the number of credentials n can be considered the VP management complexity for Idemix.

Table 7. Comparison of time complexity between SR-PP-VC and idemix

	SR-PP-VC	Idemix
VC application and issuance	$(l+2)E_1+(l+2)M_1$	$(l+6)E_1+(l+3)M_1+2H$
VC renewal	$1E_1+3M_1$	–
VP generation	$(s+6n)E_1+(s+6n-1)M_1+1H$	$(h+14n)E_1+(h+7n)M_1+2nH$
VP verification	$(s+3n+1)E_1+(s+4n+1)M_1+(n+1)P+1H$	$(s+10n)E_1+(s+8n)M_1+2nP+2nH$

This section also compares the time complexity of single credential application and issuance, update, and VP generation and verification for the SR-PP-VC scheme and the Idemix scheme, as shown in Table 7. Here, s represents the total number of attributes in each VC in the VP. The study tests and compares the management of a single credential for both schemes. In practical applications, the number of attributes in a single credential generally does not exceed 30, so tests were conducted for l = 5,10,20,30, and for h values at 0%, 20%, 40%, 60%, 80%, and 100% of l. The average time for 100 executions of single credential application and issuance, credential presentation, and verification are shown in Figure 4. When the number of credential attributes is 30, the SR-PP-VC scheme's time for single credential application and issuance is approximately 9.6 ms, the time for full attribute hiding presentation is approximately 12.8 ms, and the verification time is approximately 15.9 ms, representing efficiency improvements of approximately 35.6%, 23.6%, and 13.4% over the Idemix scheme, respectively.

(3) Contract testing

Based on the SR-DIDs and SR-PP-VC schemes, we deployed the MDIDs registration contract, DID document management contract, DIDs resolution contract, PDIDs verification contract, and DIDs revocation contract on Hyperledger Fabric. 12We used the Apache JMeter testing tool, with the number of test threads set to 10,000, to conduct average single-run time and TPS (transactions per second) tests for these contracts. The results are shown in Figure 5. The DID document management contract registers, updates, or disables DID documents on the blockchain, corresponding to the updating of VC issuer public keys in this paper.

7.3 Result analysis

Based on the results of security analysis and performance analysis, the following conclusions can be drawn:

SR-DIDs Scheme: The complexities of identity generation, verification, and regulation in the SR-DIDs scheme are all constant, reducing computational and storage overhead compared to Reference [9]. While the SR-DIDs scheme consumes an additional $2|\mathbb{G}_1|-1|\mathbb{Z}_p|$ storage space for signatures compared to the original BBS04 group signature algorithm (generally not exceeding 100 bytes on mainstream bilinear pairing curves), it optimizes the time efficiency for identity generation and verification, improving generation efficiency by approximately 48% and verification efficiency by approximately 38%.

SR-PP-VC Scheme: Compared to the Idemix scheme, the SR-PP-VC scheme optimizes credential management and performance while adding regulatory and revocation mechanisms. It reduces storage overhead by $(6n-2)|\mathbb{Z}_p|+(3n-1)|\mathbb{G}_1|$. Although the time complexity of VP generation in the SR-PP-VC scheme is $O(s)$ compared to $O(h)$ in the Idemix scheme, when the number of attributes in a single credential does not exceed 30, the overall efficiency of the SR-PP-VC scheme is superior to the Idemix scheme. Furthermore, as n increases, the SR-PP-VC scheme saves more time compared to the Idemix scheme, as shown in Table 7. For credential revocation, both the SR-PP-VC scheme and Reference [16] use accumulator technology. However, the SR-PP-VC scheme includes the credential revocation accumulator as one of the attributes signed by the issuer, allowing it to be verified along with other attributes, and reducing computational overhead. Additionally, the SR-PP-VC scheme does not require a fixed number of credential issuances, making it more suitable for practical scenarios.

PDIDs Verification Contract on Hyperledger Fabric: The throughput of the PDIDs verification contract in Hyperledger Fabric is 327, which is primarily due to the cryptographic operations involved in the verification process, as well as hardware limitations. However, in the context of regional equity markets, where the frequency of user identity management operations is relatively low and primarily focused on identity verification, credential management, and equity trading, this performance is sufficient

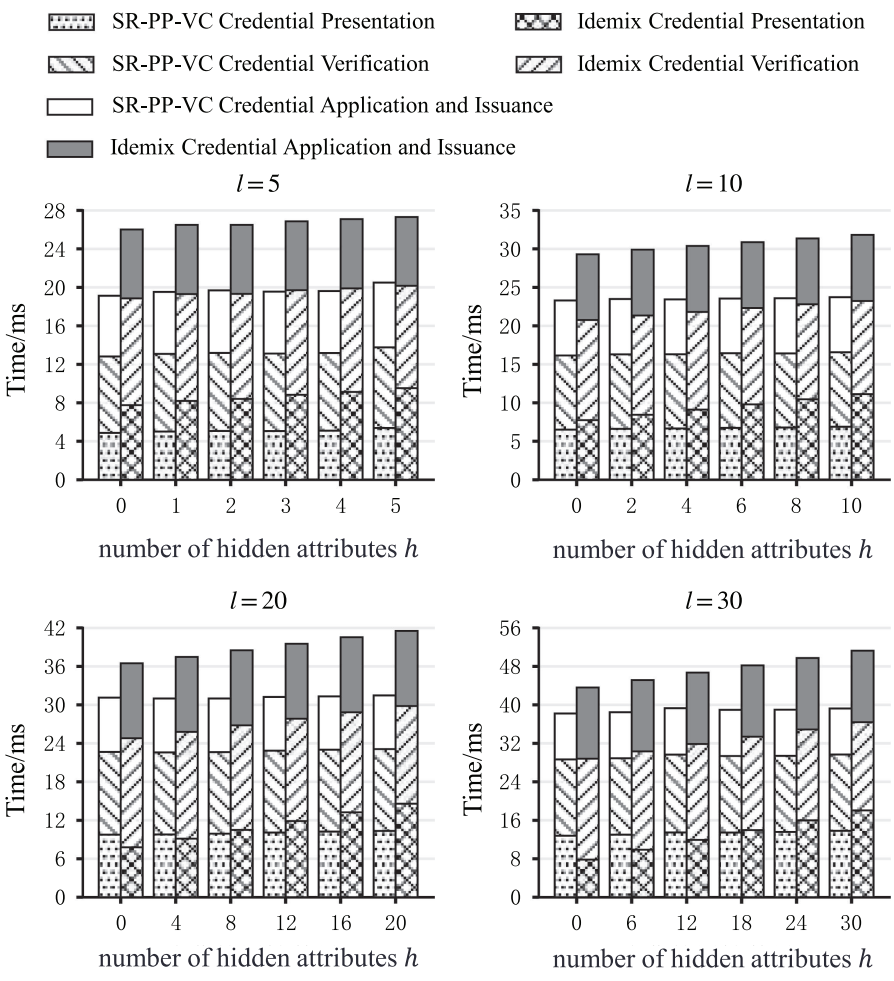


Figure 4. Time comparison of single credential management

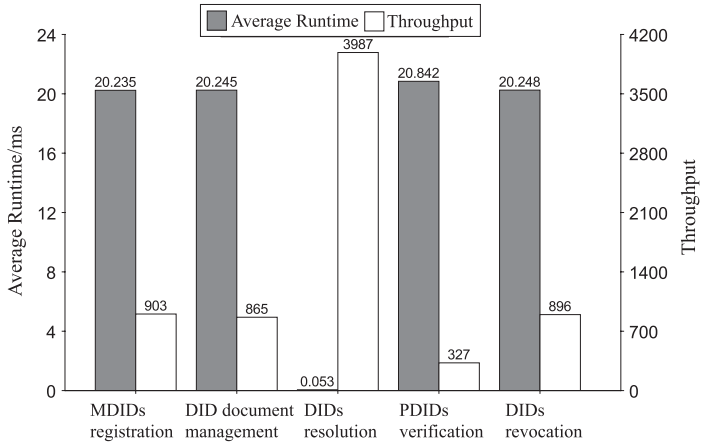


Figure 5. Test of smart contract

to meet practical demands. Furthermore, deploying the system on a production-level Fabric network would significantly enhance performance by leveraging distributed nodes, more powerful hardware resources, and optimized consensus mechanisms. The dynamic expansion of nodes and organizations would also allow the system to scale effectively, ensuring stable operation under increased loads as the market grows.

In summary, the proposed distributed digital identity privacy protection scheme offers improvements in both security and performance compared to related works. Each algorithm operates at the millisecond level, meeting the requirements for blockchain identity management scenarios.

8 Conclusion

This paper designs a blockchain-driven, regulatorily accountable, and revocable distributed digital identity privacy protection scheme. While effectively protecting the privacy of entity identities, the scheme achieves reliable supervision and revocation of identity identifiers and credentials. By optimizing cryptographic algorithms, the efficiency of identity management is improved. Future research will focus on further optimizing the VP generation scheme to ensure that the size of VPs remains constant. Additionally, the concepts presented in this paper will be used to enhance and optimize more signature algorithms, enabling VC issuers to sign VCs using various signature schemes, thereby further enhancing system security.

Acknowledgments

We thank the anonymous reviewers for their helpful comments.

Funding

This work was supported by the National Key Research and Development Program of China. Research on Reliable Regulatory Technology Based on Blockchain of Regional Equity Market (2021YFC3340600).

Conflicts of interest

The authors declare that they have no conflict of interest.

Data availability statement

No data are associated with this article.

Author contribution statement

Jing He was responsible for the design of the proposed scheme, simulation experiments, and writing the paper; Xiaofeng Ma provided guidance on the innovation and architectural design of the scheme and reviewed and revised the paper; Dawei Zhang was responsible for the feasibility analysis of the paper's proposal; Feng Peng was responsible for the verification of feasibility and experimental design.

References

[1] Sporny M, Longley D and Sabadello M et al. Decentralized identifiers (DIDs) v1.0 core architecture, data model, and representations. [2024-03-26]. https://www.w3.org/TR/did-core/

[2] Chen XF, Song ZX, Zheng PY et al. A multichain-collaborating governing chain-supervising-chain supervision framework (in Chinese). J Comput Res Dev 2024; **61**: 2290–2306.

[3] Wang Z, Chen Q and Liu L. Permissioned blockchain-based secure and privacy-preserving data sharing protocol. IEEE Internet Things J 2023; **10**: 10698–10707.

[4] Yao Q and Zhang D. Survey on identity management in blockchain (in Chinese). J Software 2021; **32**: 2260–2286.

[5] Wang C, Cheng J and Sang X et al. Data privacy-preserving for blockchain: State of the art and trends (in Chinese). J Comput Res Dev 2021; **58**: 2099–2119.

[6] Biryukov A and Tikhomirov S. Security and privacy of mobile wallet users in Bitcoin, Dash, Monero, and Zcash. Pervasive Mobile Comput 2019; **59**: 101030.

[7] AbdulKader MM and Kumar SG. A privacy-preserving data transfer in a blockchain-based commercial real estate platform using random address generation mechanism. J Supercomput 2023; **79**: 10796–10822.

[8] Li J, Wang Z and Guan S et al. ProChain: A privacy-preserving blockchain-based supply chain traceability system model. Comput Indust Eng 2024; **187**: 109831 .

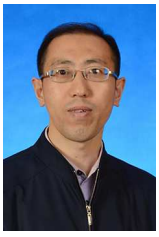
[9] Song J, Zhang D and Han X et al. Supervised identity privacy protection scheme in blockchain (in Chinese). J Software 2023; **34**: 3292–3312.

[10] Zhao L, Zhong L and Zhang J. Traceable one-time address solution to the interactive blockchain for digital museum assets. Inf Sci 2023; **625**: 157–174.

[11] Camenisch J, Mödersheim S and Sommer D. A formal model of identity mixer. In: Proc of the 15th International Workshop on Formal Methods for Industrial Critical Systems. Berlin: Springer, 2010, 198–214.
[12] Zhao Y, Yang X and Feng Q et al. Anonymous credential protocol based on SM2 digital signature (in Chinese). J Software 2024; **35**: 3469–3481.
[13] Wang Z, Fan J and Cheng L et al. Supervised anonymous authentication scheme (in Chinese). J Software 2019; **30**: 1705–1720.
[14] Boneh D, Boyen X and Shacham H. Short group signatures. In: Proc of the 24th Annual International Cryptology Conference. Berlin: Springer, 2004, 41–55.
[15] Zhu Y, Zheng H and Qin B et al. tsrCert: traceable self-randomization certificate and its application to blockchain supervision. Tsinghua Sci Technol 2023; **28**: 1128–1147.
[16] Yu Y, Zhao Y and Li Y et al. Blockchain-based anonymous authentication with selective revocation for smart industrial applications. IEEE Trans Indust Inf 2019; **16**: 3290–3300.
[17] Connolly A, Deschamps J and Lafourcade P et al. Protego: efficient, revocable and auditable anonymous credentials with applications to Hyperledger Fabric. In: Proc of the 23rd International Conference on Cryptology in India. Switzerland: Springer, 2023, 249–271.
[18] Li X, Jing T and Li R et al. BDRA: blockchain and decentralized identifiers assisted secure registration and authentication for VANETs. IEEE Int Things J 2023; **10**: 12140–12155.
[19] Manoj T, Makkithaya K and Narendra V. A blockchain based decentralized identifiers for entity authentication in electronic health records. Cogent Eng 2022; **9**: 2035134.
[20] Fukami Y, Shimizu T and Matsushima H. The impact of decentralized identity architecture on data exchange. In: Proc of the 9th IEEE International Conference on Big Data. Piscataway, NJ: IEEE, 2021, 3461–3465.
[21] Garzon SR, Yildiz H and Küpper A. Decentralized identifiers and self-sovereign identity in 6g. IEEE Network 2022; **36**: 142–148.
[22] Mukta R, Martens J and Paik H et al. Blockchain-based verifiable credential sharing with selective disclosure. In: Proc of the 19th International Conference on Trust, Security and Privacy in Computing and Communications. Piscataway, NJ: IEEE, 2020, 959–966.
[23] Maram D, Malvai H and Zhang F et al. Candid: can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In: Proc of the 2021 IEEE Symposium on Security and Privacy. Piscataway, NJ: IEEE, 2021, 1348–1366.
[24] Ran J and Cai D. Attribute signature identity authentication scheme based on blockchain and trusted execution environment (in Chinese). J Comput Res Dev 2023; **60**: 2555–2566.
[25] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review 2008; 21260.
[26] Sharifi S, Parvizimosaed A and Amyot D et al. Symboleo: Towards a specification language for legal contracts. In: 2020 IEEE 28th International Requirements Engineering Conference (RE). IEEE, 2020, 364–369.
[27] Wang D, Qin B and Song W et al. SPESC: Design and practice of smart contracts for legal purposes. Cyberspace Secur 2020; **11**: 39–46.
[28] Zhu Y, Qin B and Chen E et al. A method for transforming high-level smart contracts and the design and implementation of an auction contract. J Comput Sci 2021; **44**: 652–668.
[29] Zhang F, Hou P and Li S et al. A microservice framework for smart contracts. J Software 2021; **32**: 3423–3439.
[30] Liu J, Li P and Cheng R et al. Parallel and asynchronous smart contract execution. IEEE Trans Parallel Distrib Syst 2021; **33**: 1097–1108.
[31] Wang Z, Zhu J and Zhang B et al. Research and implementation of parallel methods for blockchain and smart contracts. Comput Sci 2022; **49**: 312–317.
[32] Galbraith SD, Paterson KG and Smart NP. Pairings for cryptographers. Discr Appl Math 2008; **156**: 3113–3121.
[33] Camenisch J and Lysyanskaya A. Signature schemes and anonymous credentials from bilinear maps. In: Proc of the 24th Annual International Cryptology Conference. Berlin: Springer, 2004, 56–72.
[34] Tessaro S and Chenzhi Z. Revisiting BBS Signatures. In: Proc of the 2023 Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham, Springer, 2023, 691–721.
[35] Chase M and Lysyanskaya A. On signatures of knowledge. In: Proc of the 26th Annual International Cryptology Conference. Switzerland: Springer, 2006, 78–96.
[36] Sean B. BLS12-381: New zk-SNARK Elliptic Curve Construction[EB/OL]. (2017-03-11), [2024-03-26]. https://electriccoin.co/blog/new-snark-curve/

Jing He is currently a master's student at Tongji University, China. His main research interests include blockchain and privacy protection.

Xiaofeng Ma is an associate professor at Tongji University, China. He is also the vice chairman of the blockchain branch of the Chinese Institute of Electronics. His main research interests include blockchain and financial technology.



Dawei Zhang is an associate professor at Beijing Jiaotong University, China. His main research interests include information security and blockchain.

Feng Peng is a senior engineer of China Securities Information Technology Service Limited Company. His main research interests include blockchain and financial technology.