Journal of Computer Science

Research Article

SEAchain: A Privacy-Preserving Dual-Credential Model on Blockchain

S. Balachander and A. Murugan

Department of Data Science and Business Systems, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu 603203, India

Article history
Received: 18-06-2025
Revised: 12-01-2026
Accepted: 19-03-2026

Corresponding Author:
Balachander S.
Department of Data Science and Business Systems, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu 603203, India
Email:
bs5678@srmist.edu.in

Abstract: All facets of daily life are now impacted by transaction data. The complex and heterogeneous network created by millions of users, devices, online services, and applications makes digital identity management more difficult. Users can maintain control over their personal information via distributed identity, a potentially useful model for resolving identity issues. Nevertheless, current state-of-the-art techniques are inappropriate because of persistent problems with transactional data and device resource restrictions, privacy and security concerns, and the absence of a systematic proof framework. In light of this, we present SEAchain in this paper as a unique distributed identity system built on a blockchain that aims to provide robust privacy preservation and self-sovereign identity. To establish a distributed identity independent of central providers, we configure transactional data to create a Sybil-resistant, unlinkable, and supervisable identity. Additionally, we implement a dual-credential model utilizing zero knowledge proofs and the Secret Encryption Algorithm (SEA) to protect the privacy of crucial attributes, on-chain identification, and credential linking. This paper introduces SEAchain, a blockchain-based distributed identity system that facilitates autonomous identity management with strong privacy protections. It eliminates dependence on centralized identity providers through a Sybil-resistant, unlinkable, and supervisable identity framework that supports light nodes. A dual-credential mechanism utilizing commitment schemes and zero-knowledge proofs safeguards sensitive attributes and onchain identity data. The system features an efficient proving process for both basic and complex credential verification. Security assessments confirm SEAchain's robustness, while experimental results demonstrate its enhanced efficiency in generating credentials and constructing proofs.

Keywords: Blockchain-Based Distributed Identity System, Systematic Proof System, Secret Encryption Algorithm (SEA), Zero-Knowledge Proofs, Privacy and Security

Introduction

A peer-to-peer, decentralised, immutable, and transparent system with append-only storage is called a blockchain. In a blockchain, transactions are organized into permanent groups known as blocks. Each block generates a hash for security, which incorporates the data, the hash of the parent block, and the current timestamp. The key feature of blockchain technology is the immutability of its records, which underpins its value. New blocks are appended to the existing blockchain through a process known as mining, and no centralised system is in control of the block (Zhang et al., 2021). The hash value is determined. Public-key cryptography has been used to guarantee the validity and integrity of transactions, and member nodes mediate them using the consensus protocol (Sharifi et al., 2021). A blockchain is defined as a developing chain of data blocks formatted through cryptographic algorithms. These blocks are interconnected to form a complex chain via the use of hash pointers (Li et al., 2021). The data's integrity can be safeguarded in this way. Blockchain enables nodes in the network to agree on the creation of new blocks through a quantifiable and verifiable process. Usually, this kind of method is called the consensus algorithm (Wang et al., 2021). Blockchain structure and consensus methods can be customised to suit specific application needs, assuming data block entanglement is guaranteed (Huang et al.,

2022). Depending on the level of openness, blockchain can be broadly divided into three categories: Consortium, private, and public blockchains (Guo et al., 2022). Public blockchains allow any node to join or leave the network freely, as they are designed to enable and document transactions on an open platform (Liu et al., 2022). Bitcoin is a prominent digital currency created and governed by a public blockchain, while the Consortium blockchain functions on a permissioned model, allowing only authorised external institutions access, with permission granted by an authorised organisation for joining nodes (Kumar and Zhang, 2022). In a private blockchain network, the nodes have mutual trust. As a result, private chains can increase efficiency by streamlining consensus algorithms and data authentication processes (Sun et al., 2022). Nakamoto initially introduced the idea of a distributed, digital, public ledger, which is now possible with blockchain technology.

It has been extensively utilised in bitcoin (Peng et al., 2022) and ether transactions. Blockchain technology is essential for various Internet of Things (IoT) scenarios, facilitating advancements through a Peer-to-Peer (P2P) network where all nodes collaborate without a central point, reducing bottlenecks. It consists of multiple blocks added over time, each containing a timestamp, nonce, hash of the previous block, and several transactions (Alharbi and Zohdy, 2023). The user data for the blockchain is captured in transactions, which are then shared with other nodes. A block is formed by some nodes gathering fresh transactions (Ahmed and Kim, 2023). A particular consensus mechanism determines how a block gets added to a blockchain. Only when every transaction in the block is legitimate will nodes accept it. Under certain security presumptions, a block cannot be altered once it is put to the blockchain. Every node continues to work on expanding the blockchain, which prevents forking. In this paper, we propose SEAchain, a blockchain-based privacy preserving scheme for SEA (Secret encryption algorithm). The majority of existing schemes don't give security and privacy much thought (Kang et al., 2023; Ferrari and Cinque, 2023; Yang et al., 2023). Some approaches ignore the need for both plaintext and cryptographic credentials, just considering anonymous credential systems. Since the system includes plaintext credentials, it's vulnerable to linkage attacks, where an adversary can infer a user's true identity from non-sensitive properties of certain credentials. To prevent this, it's crucial to safeguard sensitive credential attributes, on-chain identity data, and attribute linkages across credentials (Zhao et al., 2023; Zhou et al., 2024; Hsu and Chen, 2024). This linking issue is not present in the UTXO model's public blockchain since each transaction can create a unique identity (Nguyen and Hu, 2024). The linking problem may be resolved by a single user creating many identities, however doing so opens up the prospect of Sybil assaults in a consortium chain. Furthermore, identities must be distinct in a variety of contexts, including voting systems. As a result, it is difficult to strike a balance between linking and Sybil assaults (Wang and Lin, 2024). Furthermore, the majority of these systems fail to take IoT device and user supervision into account. In several schemes, including Know-Your-Customer (KYC), Users must provide their real names for accountability purposes, which seems to go against privacy. As a result, it is difficult to give users both privacy and supervision (Ryu and Lee, 2024). In SEAchain, Users can upload the collected data into SEAchain on a regular basis and publish it as a transaction. In actuality, Large-scale data will continue to grow due to the transactions' accelerated expansion. Users' whole data should not be stored on the blockchain because doing so will result in very high resource needs for each node. In the absence of these, the blockchain will be too difficult to search, maintain, and validate. To ensure high integrity in data storage and resilience, we introduce SWARM, a content-addressable distributed file system, in light of each blockchain node's restricted storage capacity. With SWARM, data is dispersed and stored across several swarm nodes on the Internet rather than on a single central server. Thus, there isn't a single point of failure in SWARM. Large volumes of data can be distributed effectively and duplicate-free with SWARM. Every file that is posted to the SWARM system can be recovered using a distinct hash string. The SWARM storage system in our planned SEAchain stores all of the users' data. To ensure data integrity and mapping in swarm storage, a hash string of the data stored in the blockchain is used. SEAchain provides strong scalability and supports large-scale data in this fashion. However, one of the biggest obstacles is not just the enormous amount of data that needs to be stored, but also the problem of data security and user privacy. On the one hand, user privacy is easily undermined due to the blockchain's open and transparent nature. However, you must have access to user data. Users' data should therefore be encrypted, and over the encrypted data, fine-grained access control should be implemented. SEAchain gives customers the ability to add, remove, and update authorised providers at any moment to improve data security and privacy protection.

Intuition of the Idea

SEAchain requires the protection of three types of data: On-chain identity data, sensitive credential attributes, and the connection between credentials. To ensure privacy, a dual-credential model is proposed that utilizes both plaintext and encrypted credentials. Cryptographic credentials are confirmed using zero-knowledge and encrypted using a commitment scheme without revealing the attribute values. On-chain

identification data privacy is likewise maintained by the commitment uplinking approach. Nevertheless, the attribute correlation between various plaintext credentials cannot be concealed by the aforementioned techniques. This issue might be resolved if a single person created many identities, but doing so also increases the risk of Sybil assaults within a consortium chain. We create a distributed identification system with many pseudonymous user IDs and a distinct master ID in order to resolve this contradiction. SEAChain regards master ID as a sensitive attribute, accessible to supervisors but concealed from the general public. To alleviate privacy concerns linked to attribute linkage, it employs multiple user IDs alongside a unique master ID to thwart Sybil attacks. The system tracks the master ID and relevant Know Your Customer (KYC) data while maintaining user anonymity through zero-knowledge techniques. Furthermore, SEAChain incorporates fundamental credential proofs within a methodical distributed proof system to tackle complex issues and broad assertions. This leads to the development of a distributed proof system, an identity framework, and a dual-credential model. Using several user IDs to handle privacy concerns related to attribute connections, the system utilizes a single master ID for monitoring and preventing Sybil attempts. The architecture secures sensitive credential attributes, on-chain identity data, and the connections between cryptographic credentials using commitment and zero-knowledge proofs. A distributed proof system has also been formulated to effectively express both generic and intricate credential claims

The paper presents a dual-credential privacy-preserving model designed to protect credential linkage, on-chain data on identities, and sensitive information. This model incorporates both cryptographic and textual credentials, utilizing pledges and ZKP to obscure key characteristics and identification data on-chain. Additionally, user IDs generated in the distributed identity system are used to sever connections between different credentials. A systematic distributed proof system organizes these dispersed credentials for multiconditional verification, while fundamental credential proofs are combined to solve increasingly complex problems. The SEAchain structure distinguishes between data publishing and transaction activities, employing encryption and storage via SWARM to enhance privacy while minimizing communication and computation overhead.

Related Work

The swift advancement of blockchain technology has prompted academics to investigate its potential applications in managing personal data and safeguarding privacy. Bai and Xu (2024) utilised blockchain technology to safeguard individuals' privacy over their personal information, empowering them to assume complete authority over their data (Kou and Li, 2024). The Bitnation identity registry, which is built on Ethereum and offers Estonian passports, driver's licenses, and other public facility services. Based on public Ethereum, Uport is another autonomous identity system (Rahman and Hossain, 2024). Unfortunately, due to their limited throughput and challenges in monitoring unscrupulous users, all of these public blockchain-based initiatives fall into this category (Pillai and George, 2024). Regarding permission blockchain, Sovrin is a distributed self-sovereign identity framework built on the Hyperledger Fabric that enables identity interoperability between apps and ledgers as well as decentralised identity generation on the blockchain (Khan and Malik, 2024). Users can exchange data with each other through multicentric distributed identification systems like WeIdentity and ShoCard. All the same, these works are not appropriate for the Internet of Things, and they hardly address the resource constraints and privacy and security concerns of IoT devices (Jia and Liu, 2024). The basic idea behind blockchain was to create a distributed public ledger of all Bitcoin transactions. Subsequently, numerous research endeavours concentrate on pivotal issues concerning the blockchain technology itself, like enhancing performance by resolving the double-spending threat and developing effective distributed consensus processes. Numerous other studies are also being conducted in the interim with the goal of creating useful blockchain applications. Apart from serving as the foundation for cryptocurrency systems, it may also be incorporated into several Internet of Things situations (Almeida and Costa, 2024). For instance, Chen and Zhang (2024) proposed a decentralised trust management system based on blockchain techniques to update and publish the trust information of all the vehicles in vehicular networks, enabling the effective evaluation of the trustworthiness of environments. Vehicles Additionally, in they untrusted enhanced distributed consensus by putting up a fresh method of competing for updated confidence among all RSUs. As opposed to Patel and Sharma (2024) who shared and stored vehicle data using smart contracts for effective automated data management. Ali and Hussain (2021) proposed a blockchain-based privacy-preserving and effective data aggregation scheme for smart grids, where users are divided into groups and a user is chosen as a miner to aggregate the data in each group and add it to the group's private blockchain. This allows for optimal scheduling while protecting users' private information. These plans, however, can directly solve the problems they identify in the particular network situations; yet, they are not suitable for immediate implementation in smart healthcare systems. Not only should user IoT data be protected in smart healthcare, but doctor diagnoses should also be safeguarded for privacy. From the participants' point of view, in particular, even though the user can be anyone, the doctors who diagnose users must be qualified

in order to guarantee their safety. To safeguard privacy in smart healthcare, we therefore suggest Healthchain, which consists of a Userchain and a Docchain. Singh and Gupta (2024) suggested a decentralised anonymous credential method for the traditional Public Key Infrastructure (DPKI) that permits identity assertion while maintaining privacy and does away with the necessity for a centralized one that allows provers to selectively reveal their credentials to verifiers, was proposed by Yuan and Tang (2024). But in order for U-Prove to request claims, the issuers must communicate with the company online at all times. They are still far from a distributed identification system that is self-governing, though, as they are merely systems for issuing individual credentials. A proof method was developed for general and sophisticated assertions, establishing knowledge of the constituent parts of any knowledge specification set; however, it is a centralized proof system (Lopez and Perez, 2024). An access module (Taccess) and a data storage module (Tdata) are both components of the blockchain-based personal data management system that Zyskind et al. (2015) have proposed. This system is designed to protect users' privacy. Through the use of protocol transactions, Taccess makes it possible for data owners to establish a variety of access policies for authentication methods. This enables a dynamic and fine-grained access control system. Additionally, it features a distributed storage system utilizing blockchain technology to ensure fine-grained access control based on attributes (Qin and Li, 2024; Rasheed and Salah, 2024). Nevertheless, these efforts are unsuitable for cross-system and cross-application scenarios in a distributed environment, as they are intended for singlecloud settings (Jiang and Han, 2024). Moreover, the significant overhead hindered their suitability for Internet of Things situations.

System Overview

System Model

In our system paradigm, SEAchain involves issuers, users, verifiers, and supervisors. Users must select at least t nodes to validate transactions related to credential services. We choose for a consensus blockchain, in which devices can serve as verifiers and the committee can create credentials. Constrained devices are deployed as light nodes, and issuers, like committees, are configured as complete nodes. While the verifier verifies the credentials and publishes the access policy, the text indicates that an issuer is responsible for both signing and issuing credentials. This involves the formal authorization and dissemination of documents or digital credentials by the issuer, serving as a verification of identity or qualifications. The process of signing implies a confirmation of authenticity, while issuance indicates the distribution of these credentials for use. The person in charge of confirming the user's actual identification and conducting audits is the supervisor. The person wishing to access the application is the owner of the credentials. Every user has two DIDs (masterID and userIDs). The userID is used for several daily applications, while the masterID is specific for supervision. Every user possesses a public-private key.

Security Model

The Byzantine failure model introduces an adversarial framework permitting committee nodes that are malfunctioning or attacking, where $t \rightarrow N/3$. This model operates under an asynchronous communication paradigm, accommodating erroneous, delayed, or undeliverable messages.

In terms of security properties within the identity system, three key features are achieved: Supervisibility, unlinkability, and Sybil resistance. The concept of a Sybil attack is defined as a scenario where a limited number of entities create multiple peer identities to amass greater shares of the system. Unlinkability ensures no identifiable relationship between two distinct identities, maintaining their separation despite the attacker's prior knowledge gained through observation. Supervisibility indicates the ability of the supervisor to swiftly locate essential information accounts and real-world identities to extract necessary information during oversight.

Furthermore, the credential system is designed to be private, unforgeable, and unlinkable. Unforgeability guarantees that an attacker cannot replicate the current credentials of legitimate users to produce fraudulent credentials. Unlinkability prevents an adversary from associating a credential with multiple user submissions or transactions, thus obscuring user characteristics or identifiers. Lastly, the privacy aspect ensures that the process of issuing and validating credentials keeps user traits undiscoverable by adversaries.

Cryptographic Credentials

Commitment Schemes: Sensitive identifying information, like an ID card, cannot just be revealed in plaintext. To preserve the privacy of one's identifying characteristics, we reveal pledges to the claim(s) to keep personal values hidden. The Pedersen commitments method is specifically used by SmartDID. Let g and h have two random generators, and the text provides an explanation of the Pedersen commitment by situating it within the framework of a cyclic group G of order 1. In order to commit a secret integer value v, which is located within the range 0,1, q 1, a randomization value r can be utilized during the process. A representation of the commitment can be found in the equation com: = Comm (v,r) = gvhr. It is highlighted that the randomness r provides a perfect guarantee that the commitment will be kept hidden during the entire process. Nevertheless, if the randomness r

is determined to be zero, the commitment will continue to be enforceable, but it will not be concealed.

Pedersen Vector Commitment G has two random generators: g = g1, 0020gn Gn and h G. The Pedersen commitment can thus be calculated with a secret integer vector v = v1, vn as follows: com: = Comm (v, r) = gvhr = hr gvi G with the randomness r. Assuming perfect hiding and discrete logarithms (DLs), Pedersen vector I commitment is computationally bound.

Zero-Knowledge Proofs: SEA Chain uses ZKP to safeguard the privacy of credentials. In order to accomplish a mission, two or more parties must follow certain procedures. In general, the person who possesses the secret is the prover, and the person who is persuaded to believe it is the verifier. For instance, it is possible for a prover to be aware of the openness of the commitment com and to have the intention of persuading a verifier that he completely comprehends the committed value v, such as v = 100. Through the utilization of noninteractive bulletproofs, the prover has the ability to generate a proof that successfully convinces the verifier without divulging any information regarding v. The prover and verifier do not communicate with each other while using the hash method. The prover can upload the credentials to the decentralized ledger and add to them to enable verification by any proof system.

Range Proofs for Bulletproofs: Similarly, suppose a prover knows the opening of the commitment com and wants to convince a verifier that the committed value v is inside a specific range, like 0 v ¡ 103. The prover then generates a Range proof and adds it to the credential and distributed ledger so that any proof system can validate the credentials via blockchain.

Decentralized Identifier

The following is a definition of the DID syntax: The variable name" did" is defined as" did:did-method:did-method-specific-id," where" did" is a fixed string: Is used to combine strings, and" did-method" and" did-method-specific-id" are their respective names. A DID document is connected to each individual user ID, that makes the user's public key and authentication method available to the public.

1. Verifiable Credentials: Credentials comprise credential metadata, claims, and proofs that adhere to in the context of the World Wide Web, the notion of verified credentials
2. Metadata: The metadata includes conventional items such dates of issuance and expiration. The metadata will be removed later for simplicity
3. Claim(s): A user-issued statement is the claim. The expressions claim = att, com, U and claim = att, val, U can be used in attribute-value-user relationships and attribute-commitment-user relationships, respectively. Users' identities are denoted with U
4. Proof (s): The zero-knowledge proof or digital signature of the credential is the proof

Dual Credential Model

SEA Chain

The distributed identity system, among the components that make up an identity management approach are the dual-credential model and the systemic distributed proof system. (Setup, Create DID, Create Claim, Create Credential, Ver Credential) is how we defined SEA chain.

Following the receipt of the security parameter, the issuer proceeds to construct q-order cyclic groups G by employing generators of either (h, g) or (h, g). This configuration is referred to as sp (Ahmed and Kim, 2023). A bilinear map is represented by the expression e: G × G → GT. Sp is the parameter, and it is equal to g, q, h, g, G, TG, e, and H. The signature scheme, which can be either Sign or VerSign, and the collision-resistant SHA-256 hash algorithms H: 0, 1 → qZ should be chosen.

CreateDID(sp) → (pkU, skU), DID: Once the method receives the system parameter sp, it performs the generation of a public–private key pair (sk, pk) and the identifiers DID = (masterID, userIDs).

CreateClaim (val, att, U) → (Claim, U): Once the attributes (val, att) are obtained from user U, the algorithm generates the claim as claim = claim1, claimm together with a signature U = Sign (claim, skU), where claimi is equal to val, att, U

A consensus is reached by the committee nodes once they have received the user's public key (pkU), the issuer's private key (skI), the credential claim, and the Shamir threshold scheme. They then verify the accuracy of the claim and construct a signature in the form of a signature. This process is referred to as the CreateCredential (skI, pkU, Claim, U,) → (Cred, Proof) algorithm. The issuer generates the credential by using the formula cred = claim, pkU, and proof = (I, dg), where dg is a hash or commitment digest of cred and I = SignpkU, (skI, dg). Once the verification of U has been completed, the issuer generates the credential.

B: VerCredential (ch, pkI, cred, with an E) Following the acquisition of the user identification DID, the issuer public key pkI, the tree structure E, the credential cred, and the challenge ch (in order to prevent replay attract), the verifier computes the Boolean value of the tree proof system and verifies the authenticity of each credential. This is done in order to prevent replay attract. The result of the method would be b = 1 if all of the verifications were successful, and b = 0 if they were not successful.

It is important to note that SEAChain updates and revokes credentials in accordance with the PKI infrastructure system.

System of Distributed Identity

We go into great detail on the distributed identity system in SEAChain. unlinkability, Sybil resistance, and supervisibility are the identity system's overarching objectives.

Unlinkability and Sybil resistance appear to be mutually exclusive. An adversary has the potential to collect various attributes, which are not necessarily sensitive, from multiple credentials associated with a specific identifier. This risk arises due to the presence of certain plaintext credentials within the system. This would allow them to deduce the user's true identity. Because Bitcoin can create a new identity for every transaction, it does not have this issue. A consortium chain may be vulnerable to Sybil assaults, even though a single user may be able to overcome the linking issue by creating many identities. To compromise a disproportionate percentage, A single user can create and manage multiple identities, which allows them to execute Sybil attacks effectively. This tactic involves using these false identities to manipulate and exploit systems, potentially leading to detrimental impacts on network security and integrity. To address this issue, we create a distributed identification system that uses many pseudonymous identifiers (user IDs) and a distinct masterID.

In a similar vein, maintaining privacy and supervision is challenging. Identity privacy is in conflict with supervision since it requires traceability to the user's true identity. SEAChain addresses the challenges posed by Sybil attacks and linkage attacks through the implementation of a mapping table that associates master IDs with user IDs. The user's safety is improved by this forward-thinking strategy, which also identities but also fosters user supervisibility and privacy. By leveraging a combination of commitment schemes and zero-knowledge proofs, SEAChain effectively protects users and reinforces the integrity of their personal information within the system.

Identity Registration: In accordance with the W3C DID standard, an identity system is created by SEAChain, which includes a number of pseudonyms, also known as user IDs, as well as a master ID. The master key, which is unique to the entire system, operates in a manner that is analogous to that of a database. With r being a random number that changes depending on the credentials, the master ID is represented by the Pedersen commitment to network, which is written as com = Comm(r,v) = hrgv. A public-private key pair is associated with each of the several user IDs that are possible.

DID issuance and registration for users are based on blockchain consensus. In the PBFT (Practical Byzantine Fault Tolerance) consensus model, there are three distinct types of nodes: The client, the master, and the replica. Each serves a specific function within the consensus process, contributing to the overall efficiency and reliability of the system. Sending transaction requests is the responsibility of the client node. In the consensus mechanism described, a single primary node is accountable for the process of integrating transactions into blocks and supporting the process of reaching consensus on blocks during each round. Alongside this primary node, several replica nodes participate in the block consensus process, each performing data processing in a similar manner. Collectively, both primary and replica nodes constitute the consensus nodes involved in the protocol. The Practical Byzantine Fault Tolerance (PBFT) consensus process consists of three distinct steps: The first step is to prepare, the second step is to prepare again, and the third step is to commit the transactions. This structure ensures a robust framework for achieving agreement among distributed nodes in the presence of faulty components.

The steps involved in user DID registration are: Initially, the client submits a master ID registration request; next, the system applies the consensus algorithm; third, in the outlined process, the system initially addresses the client's request as the first step. Following this, the client verifies that a consensus has been achieved. Subsequently, the client proceeds to submit a user ID registration request, thereby initiating a new consensus round. These steps encapsulate the pertinent details of the procedure. System Consensus Using PBFT: This is the first stage in the process that has been outlined, and it involves the system responding to the request made by the customer. The next step is for the customer to confirm that they have reached a consensus on the matter. Following then, the client will proceed to send in a request for user ID registration, which will result in the beginning of a new round of consensus. This technique is broken down into these steps, which contain all of the important elements.

The client checks to see if the system has agreed to his request after receiving the commit messages. If so, the client will receive a response from the system with the master ID.

After receiving the response, the client verifies that the consent has been fulfilled and discreetly stores the master ID.

In order to start a new consensus round, the client submits the master ID commitment and keeps applying for user IDs.

The user ID is public to the network, which is the difference and more thorough PBFT procedure. In the process of identity verification for SEAChain, the master ID serves as a unique identifier that users must present to their supervisors along with an associated random number.

Theoretically, different random numbers are recommended for various credentials to enhance security; however, managing these numerous random numbers can complicate the system, potentially hindering the supervisor's ability to access the corresponding commitments seamlessly. To streamline this process, users are required to provide a public verifiable token, denoted as tk = (pk)r, which allows the supervisor to validate identities without needing to keep track of the random number. The master ID is encrypted with the commitment formula com:= Comm(r,v) = hrgv. By revealing the token, the user enables the supervisor to open the commitment conveniently, thus simplifying the identity verification process while maintaining security integrity.

To establish that com = hrgv represents an open commitment to v, where v represents the value of master ID, a supervisor has the capacity to compute tk = pkr = hskr = ssk and s = com/gv = hr. This is the case in the event that the supervisor wishes to demonstrate that com = hrgv represents an open commitment to v. In order to fulfill their duties, the supervisor is only required to provide evidence that logstk is equivalent to loghpk. This illustrates that both logarithms are computed as sk, and that the entire equation is independent of r due to the fact that pk equals hsk. Moreover, this demonstrates that the logarithms equal sk. Because of this, the supervisor is excused from the requirement to provide this documentation.

Likewise, batch verification is supported by SEAChain and is represented by the formula comk = g vkh rk. Because of the DL issue, supervisors can only use the public tokens to confirm their commitments; malevolent users are unable to discover any personal information about other users.

The plaintext claim and commitment claim are matched by the cryptography and plaintext credentials in our system. With the use of blockchain's consensus mechanism, SEAchain boasts the following characteristics:

- An attribute that does not exist cannot be created by the user
- To successfully complete the verification, the user must possess the given properties

Hiding Attributes

Plaintext and commitment properties are both supported by SEAchain. SEAchain encrypts the properties in the claims using Pedersen commitments in cryptographic credentials. Presuming that a pledge is opening to the value:

$$Enc := Enc_c(u,r) = D^r B^r$$

For an attribute u, or a commitment vector is allowing:

$$Enc := Enc_c(u,r) = D^r B^r$$

For an attribute vector u to open. The commitments of attributes are therefore completely indistinguishable, making it impossible for an opponent to determine whether a value is positive, negative, or zero. If it is required, the user can show the value r and randomness u to a verifier who is familiar with the commitment com, and the verifier is able to confirm that they are consistent.

Building Claims

Let vector $TA = \{TA_1, TA_2, TA_n\}$ represent the corresponding user attribute values associated with user V. Assuming a credential vector $PS = \{PS_1, PS_2, \ldots PS_n\}$ contains m claims, the system maintains a set of claims:

$$H(.) = \{H(.)_1, ..., H(.)_M\}$$

Where $H(\cdot)$ consists of both commitment claims and plaintext claims.

Plaintext claim: The values for characteristics in a plaintext claim are represented in the form:

$$H(.)_i = \{PS, TA, V\}$$

Commitment claim: In a commitment claim, the values are expressed as:

$$H(.)_i = \{PS, Enc, V\}, \text{ where } Enc = Enc_c(TA, r)$$

Here, Enc_c (TA, r) denotes the Pedersen commitment to attribute vector TA using a random method r.

Finally, the user U creates a digital signature as proof that the assertion is true using their private key DKU, given by:

$$S_U = \text{si}\,gn(DK_U, H(.))$$

Creating Credentials

Prior to the credential proofs being uploaded to the blockchain, consensus is conducted to ensure the legitimacy and dependability of the credentials. Generally, some issuers are organizations possessing a certain level of expertise regarding the user.

Assume that there are at least t nodes participating in a (t, N) Scheme of the Shamir threshold Ψ to support the assertion. Each committee member C_j reaches an agreement, creates a signature in the following manner after verifying the assertion:

$$\sum_j = sign(DK_j, H(.))$$

In order to give evidence that the assertion is correct. The final step is for the user to join all of the different signatures into a single integrated signature:

$$\sum_{j} = \left\{ \sum_{1} \| \ldots \| \sum_{N} \right\}$$

Where ‖ denotes the concatenation of the individual signatures. Next, the issuer creates a credential defined as:

$$Hash(.) = \{EKv, H(.), \varepsilon\}$$

Where ε is the signature evidence of the claim. Let (EK_I, DK_I) and (EK_U, DK_U) denote pairs of public and private keys belonging to the user and the issuer, and vice versa.

The issuer then determines a proof of commitment as:

$$\sum_{1} = sign\left(DK_1, EKv, \varepsilon, g_q(.)\right)$$

Where gq represents the commitment value or hash of the credential. SEAchain supports numerous cryptographic credentials, including AND and OR credentials.

Discrete Logarithm (DL) Proof

The discrete logarithm (DL) proof is a cryptographic primitive used to demonstrate the existence of a specific private value. It is foundational for constructing more complex proofs such as and OR proofs.

We demonstrate that:

$$Enc = D^r E^y, where\ Enc = D^{v-Ch.x}.D^x.Ch = Enc$$

And the corresponding challenge hash is:

$$Ch = B(D, y, Enc) = Ch$$

AND Proof

Building upon the DL proof, the AND proof allows a prover to demonstrate that they simultaneously possess multiple attributes.

We show the accuracy of:

$$Enc_1 = D_1^r E_1^{y1}, Ch_1 = D_1^{u1-Ch.x_1}.D_1^{x1}.Ch = D_1^{u2} = Enc_1$$

$$Enc_2 = D_2^r E_2^{y2}, Ch_2 = D_2^{u2-Ch.x_2}.D_2^{x_2}.Ch = D_2^{u2} = Enc_2$$

Thus, the combined challenge hash for the and proof is:

$$Ch = B(D_1, y_1, D_2, y_2, Enc_1, Enc_2) = Ch$$

OR Proof

If the prover is required to demonstrate that they satisfy at least one attribute set (logical OR), an OR proof is constructed, as outlined in Algorithm 3. We demonstrate:

$$Enc_1 = D_1^r E_1^{y1}, Ch_1 = D_1^{u1,y1} = Enc_1$$

$$Enc_2 = D_2^r E_2^{y2},\ Ch_2 = D_2^{u2-Ch2.x_2}.D_2^{x_2} = Enc_2$$

The or challenge hash is calculated as:

$$Ch = B(D_1, y_1, Enc_1, Enc_2) = Ch$$

And satisfies:

$$Ch_1\, Ch_2\, a + (Ch - a) = Ch$$

Uplink Strategy

Credential uplinking methods are developed by SEAchain in order to reproduce the commitment claim and the plaintext claim. These algorithms are designed to meet a variety of security requirements for hash and commitment uplinking.

Hash Strategy for Plaintext Credentials

The system falls back to employing a hash technique for uplinking in the event that when it comes to the user attributes on the chain, there is a minimal demand for privacy restrictions.

Algorithm 1 To demonstrate that A is the discrete logarithm of the fraction B to begin with g, use secret A and generator g:

Input: Secret A, generator g
Output: Challenge CX, response RY

```
1: function CREATE DL PROOF(a,g)
2:         B ← g^a
3:         u ← random ∈ Zq
4:         com ← g^u                          ▷ // commitment to u
5:          cx ← H(g,y,com)
6:           ry ← u−cx·a mod q
7: end function
8: function VERIFY DL PROOF(cx,ry)
9:         Enc′ ← gryycx
10:          flag ← cx =? B(g,y,Enc′)
11:        return flag
12: end function
```

Given a credential represented as:

$$Hash(.) = \{EKv, H(.), \varepsilon\}$$

Where EKV is the user's public key, $H(\cdot)$ is the claim, and ε is a proof signature. The credential's hash value is then signed by the issuer, producing the proof as:

$$\sum_{1} = sign\left(DK_1, EKv, \varepsilon, g_q\right)$$

Where: - *DKI* is the issuer's private key, - *EKV* is the user's public key, - ε is the signature evidence, - gq is the credential digest or commitment value derived from the hash.

Following that, the proof that was generated is uploaded to the blockchain through the use of a smart contract in the following format:

$$\text{Proof} = \{g_q, \varepsilon_I\}$$

This ensures that the credential and its verification data are securely registered and tamper-proof within the blockchain infrastructure.

Commitment Strategy for Cryptographic Credentials

The low entropy that arises from the limited number of attribute values has the potential to render the hash vulnerable to traversal attempts, which in turn could expose the personal information of the user. Due to the fact that the hash only holds a limited amount of properties, this is the case. As a result of this, we now upload the commitment rather than the credential hash value in order to ensure the integrity of the digest data that is kept on the chain. This is done in order to guarantee the safety of the data. There is a distinction between the digest method and the hash strategy for plaintext credentials; the digest method is the source of the discrepancy. The commitment value gq is computed as:

$$g_q = Enc_c\left(Hash(.), r\right)$$

Where: - $Enc_c(\cdot)$ denotes the commitment function, - *Hash (·)* is the digest of the credential, - *r* consists of a random number that serves as the blinding factor for the commitment.

Distributed Systemic Proof System

Tree-Based Distributed Systemic Proof System

It is through the utilization of a tree-based distributed systemic proof system that we develop a logical access structure. Credentials are incorporated into the leaf nodes of the structure. The implementation of attribute-based fine-grained access control is carried out via this structure and serves as an extension of Public Key Infrastructure (PKI).

Each of the tree's nodes that is not a leaf is defined by its child nodes, which can represent logical structures such as AND-gates or OR-gates. We denote the set of supported operations as:

$$OP = \{\cap, \cup\}$$

Algorithm 2 To demonstrate that B_1 the discrete logarithm of is the symbol A_1 to begin with g_1, and B_2 the discrete logarithm of is the symbol A_2 to begin with g_2, there is a secret A_1,A_2, generator g_1,g_2.

Input: Secret A_1,A_2, generators g_1,g_2
Output: Challenge cx, responses rp_1,rp_2

1: **function** CREATE AND PROOF(A_1,A_2,g_1,g_2)
2: $B1 \leftarrow gA11$, $B2 \leftarrow gA22$ 3:
$u_1,u_2 \leftarrow$ random $\in Zq$
4: $Enc_1 \leftarrow g^{u}{}_1{}^1$, $Enc_2 \leftarrow g^{u}{}_2{}^2$ ▷ // commitments 5: cx ← $B(g_1,B_1,g_2,B_2,Enc_1,Enc_2)$
6: $rp_1 \leftarrow u_1 - cx \cdot A_1 \bmod q$, $rp_2 \leftarrow u_2 - cx \cdot A_2 \bmod q$
7: **end function**
8: **function** VERIFY AND PROOF(cx,(rp_1,rp_2))
9: $Enc1' \leftarrow g_1^{rp_1} B_1^{cx}$, $Enc2' \leftarrow g_2^{rp_2} B_2^{cx}$
10: flag$\leftarrow cx \stackrel{?}{=} B(g_1, B_1, g_2, B_2, Enc_1', Enc_2')$
11: **return** flag
12: **end function**

Additionally, attribute weights may be assigned to influence evaluation order or priority. After performing operations on its child nodes, a non-leaf node evaluates to a Boolean value. Specifically:

- An AND-gate (∩) requires that both right and left child conditions be satisfied
- An OR-gate (∪) requires that at least one of the child nodes be satisfied

Construction of the Access Tree: Consider the following definition for the set of credential relationships:

$$E = \left(Hash(.)_0 \cap Hash(.)_1\right) \cup \left(Hash(.)_3\right) \cap Hash(.)_4$$

And Let the values of the corresponding weights be:

$$W = \{0.5, 0.4, 0.4, 0.4, 0.9\}$$

To optimize tree construction and avoid recursive insertion inefficiencies, the credentials are first sorted by descending weight. Credentials with higher weights are assigned to the left child nodes. This ensures in the left subtree, the maximum weight is higher than the maximum weight in the right subtree.

For example, as shown in Figure 4, we have 0.5 > 0.4. The weights for the left node of the subtree L1.100 and right subtree node L1.101 are 0.5 and 0.4, respectively.

The root node of the access tree is designated as Level 0 (denoted as L0). Labeling continues sequentially for its left and right child nodes as L1.0 and L1.1, and so forth.

Algorithm 3 To demonstrate that B_1 the discrete logarithm of is the symbol A_1 to begin with g_1, or B_2 is the discrete logarithm of A_2 with base g_2, use secret A and generators g_1,g_2.

Input: Secret A_1,A_2,g_1,g_2
Output: Challenge cx,cx_1,cx_2, response rp,rp_1,rp_2

1: **function** CREATE AND PROOF(A_1,A_2,g_1,g_2)
2: $A_1 \leftarrow$ random $\in Zq$, $A_2 \leftarrow A$
3: $B1 \leftarrow gA11$, $B2 \leftarrow gA22$
4: u_1,u_2,$p \leftarrow$ random $\in Zq$
5: $Enc_1 \leftarrow g^{u_1 1} B_1^{p}$, $Enc_2 \leftarrow g^{u_2 2}$ ▷ // commitments
6: $cx \leftarrow B(g_1,B_1,g_2,B_2,Enc_1,Enc_2)$
7: $cx_1 \leftarrow p$, $cx_2 \leftarrow cx - cx_1$
8: $rp_1 \leftarrow u_1 \bmod q$, $rp_2 \leftarrow u_2 - cx_2 \cdot A \bmod q$
9: **end function**
10: **function** VERIFY OR PROOF(cx,cx_1,cx_2,rp_1,rp_2)
11: $Enc_1' \leftarrow g_1^{rp_1} B_1^{cx_1}$, $Enc_2' \leftarrow g_2^{rp_2} B_2^{cx_2}$
12: flag$\leftarrow cx \stackrel{?}{=} B(g_1, B_1, g_2, B_2, Enc_1', Enc_2')$ and $cx_1 + cx_2 = cx$?
13: **return** flag
14: **end function**

Verification of the Access Tree

To verify an access tree, SEAchain uses a Depth-First Search (DFS) traversal strategy. The tree is composed of logical gates (AND ∩ and OR ∪), with credentials embedded in the leaf nodes.

During Verification

- If an OR-gate (∪) node receives a TRUE from either its left or right child, the verification is completed successfully for that node
- If an AND-gate (∩) node receives a FALSE from either child, the verification fails at that node
- If all checks return TRUE, the access tree is verified as satisfied

Algorithm outlines the recursive procedure for verifying access trees based on DFS traversal.

Verification of Leaf Nodes

To confirm the identity holder, each credential in a following action

The identity holder is verified by the verifier. This step verifies that the holder actually possesses the private key. The holder first issues a challenge, creates the challenge hash, and uses his private key to sign the hash value The verifier receives both the challenge and the signature. Fig. 1, depicts the Building the access tree, subsequently, the verifier uses the purpose of decrypting the signature and obtaining a hash, the public key of the holder. Following that, he employs the same hash algorithm in order to generate a new hash by making use of the initial challenge. A comparison is then made between the revised hash and the previous hash by the verifier. Only in the event that the two hash values are identical will the verification process be considered successful, proving that the private key and public key used to encrypt and decrypt the signature are compatible.

The verifier verifies the issuer's existence. If the issuer's name appears in the list of issuers, call the smart contract to find out. Subsequently, the verifier verifies the authenticity of the credential content, including the expiration date, credential format, and the issuer's signature. Actually, simply encrypting the private key of the issuer is all that constitutes a digital signature. It is possible for the verifier to decrypt the signature by utilizing the public key of the issuer and receive a credential hash or commitment. Fig. 2 verifies the access trees verification.

The credential's digest is recreated by the verifier. The holder's credential body's hash value during the process of creating a new digest and comparing the two digests, the verifier will calculate the hash value, also known as the commitment value. Except in the event that the two digests are in agreement, that is, when the official issuer validates the qualities of the holder will the verification be successful.

Additionally, the digests and the blockchain proof will be compared by the verifier via calls to smart contracts. The verification will only be successful if all three digests match. In conclusion, the algorithm yields FALSE otherwise and TRUE if all stages are completed.

Sample Access Tree for Recruitment

A sample access tree for a business that hires staff members is illustrated in Figure 3. Verifying the authenticity of candidates' identity documents, educational credentials, and personal resumes is essential in the recruitment process.

Two access strategies are defined as follows:

a) (Credential equivalent to a Bachelor's Degree ∩ Graduation Credential) ∩ID Card) ∩Resume of the Student)
b) (Credential equivalent to a Bachelor's Degree ∩ Graduation Credential) ∩Student Card) ∩Resume of the Student)

A verifier usually maintains several access policies, which he modifies in accordance with a rule tree. The SEAchain platform is capable of supporting many access policies, which may include the combination of Plaintext, AND, OR, and Range credentials.

Storage

SWARM is a decentralized storage system designed to provide distributed, fault-tolerant, and censorship-resistant data storage. It works by splitting data into chunks and distributing them across multiple nodes in the network. SWARM ensures data availability and integrity through cryptographic hashing and incentive mechanisms.

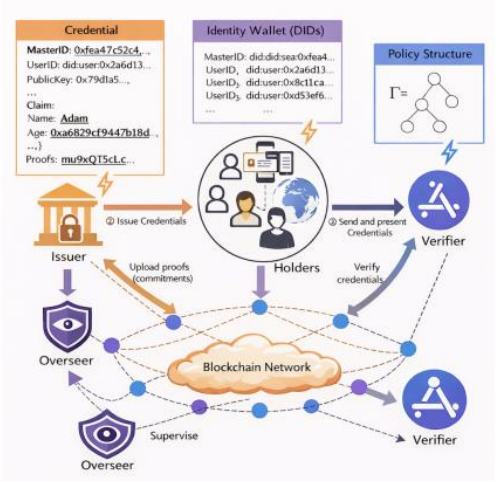

Fig. 1: System model of Sea Chain

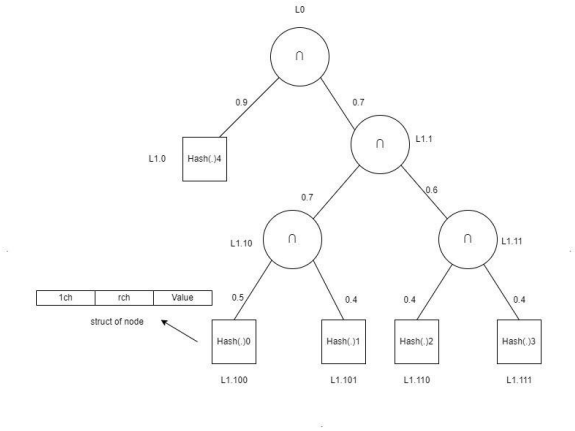

Fig. 2: Building the access tree

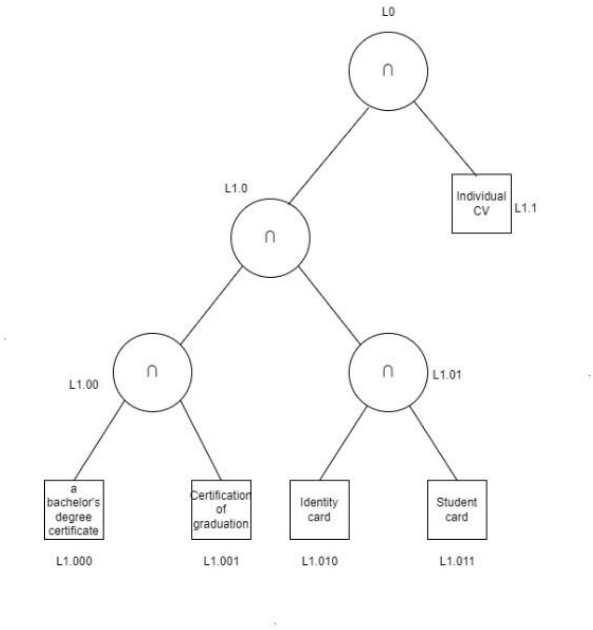

Fig. 3: The access tree's verification

It integrates efficiently with blockchain systems to support decentralized applications. Overall, SWARM enables secure, scalable, and trustless data storage without centralized control. They cooperatively store, in a dispersed fashion, encrypted diagnoses from transactions and fully encrypted data from transactions. In this work, we store all storage nodes are SWARM-based storage system, and that transactions and other collaborate to manage and maintain the swarm system. SWARM employs a content addressing mechanism in which the file's content serves as the source of the address. To uniquely identify each file, each hash string is created by hashing the file. Anyone can access the entire file saved in swarm by using the file's hash string on SEAchain. High data quantities can be distributed very effectively.

Security Analysis

In this section, we analyze the security properties of SEA chain.

Security Properties of the Credential System

Unforgeability, unlinkability, and privacy are all achieved through the use of the credential system. The completeness, soundness, and completeness are all possible and zero-knowledge characteristics of zero-knowledge proofs technology to our cryptographic credentials. The following is a study of the dual-credential model's security features.

Unforgeability

The identity wallet is the only place where the user's private key is ever stored offline, while using SEAchain's distributed identification system. In the verCredential phase of protocol execution, the user only signs a task for the verifier using the private key. In the same way, the issuer only signs the proofs' digest of claims using the private key during the create Credential phase. Furthermore, the owner who can provide proof of identification is the only one who can access the public key, and deriving the key that is distinct from the system's public key Private key is a challenging task. That is, there is no way to hack the system.

Unlinkability

Cryptographic Credential Creation and Verification the user makes use of the security parameters in order to create cryptographic credentials throughout the phase of verification and generation of cryptographic credentials:

$$Seck=\{D,r,P,r,RT,e,\ B\}$$

The attribute random:

$$Hash(.)=\{EK_1,H(.),\varepsilon\}$$
$$H(.)=\{PS,Enc,V\}$$

$$Ek_1=\{PS,TA\}$$

Where in credential ε is the claim evidence that committee nodes have signed, and:

$$Enc=Enc_c(TA,r)$$

Is the commitment to *TA*. *A* proof is then produced by the algorithm as:

$$\sum_1=sign\{DK_1\,EK_{\cup},\varepsilon,g_q\}$$

Before presenting, the prover in every verification should select a random number, and the verifier should examine the signatures of *ε, εI*. When using the zero-knowledge verification procedure, the zero-knowledge attribute makes sure that no information regarding (PS, TA).

To provide anonymity and unlinkability across various verification phrase executions, the verification phrase is used to generate a random number, which is then used to generate a random number for the credential. Furthermore, the user ID is almost randomly generated and the master ID is encrypted, making it impossible to associate a user's identity with the plaintext of a single credential or the commitment qualities.

The attributes gathered the previous time Since the master ID is encrypted and the user IDs are not, it is impossible to establish a connection to the subsequent time are produced virtually at random each time.

Credentials and identity are therefore unlinkable since an attacker cannot carry out repeated collections to carry out a linkage attack. In summary, identity, credential, and verification are not related to one another.

Privacy

Private Phases in Cryptographic Credential Construction and Verification

There are two private phases involved in creating and verifying cryptographic credentials. As we begin the credential building phase, we examine the information the adversary has gathered

When creating a claim phrase, the characteristics PS, TA information is encoded into a commitment vector of:

$$Enc=Enc_c(TA,r)$$

Where *r* is an arbitrary number, so:

$$H(.)=\{PS,Enc,V\}$$

After that, the credential is output and the issuer signs it as:

$$Hash(.)=\{EKv,H(.),\varepsilon\}$$

The issuer, or committee node, in this credential issuance procedure confirms the commitment, and it is via the committee node that the adversary obtains knowledge of the output commitment. The claim's commitment value can be discovered by the adversary

Nothing further is disclosed because the method of proof is known as zero-knowledge. and the commitment is concealed.

In the verification stage, an adversary can obtain knowledge of the output commitment by using the verifier to input the identifier SEA chain, the issuer's public key pkI, and the cryptographic credential cred. During this process, the adversary is only able to learn the commitment value of the claims, and the proof consists of the stamp of approval from the Issuer as well as the claim digest. It can only, however, confirm that the pledge and the signature are accurate. Nothing further is disclosed because the commitment is concealed using the discrete logarithm problem and zero-knowledge verification.

Implementation and Performance

The performance of the suggested strategy will be assessed and experiments will be designed in this part.

Implementation

We have developed a prototype for our identity management scheme operating on the Fisco Bcos

blockchain, utilizing limited IoT devices and users functioning as light nodes, alongside Bulletproofs, a versatile proof system. SEA chain, our distributed identity system, comprises three main components: the distributed identity system, the nested distributed proof system, and the dual-credential model. Cryptographic credentials are generated through various methods, including Pedersen commitment, standard zero-knowledge bulletproofs, and range proofs, enhanced by additional AND and OR proofs.

In our implementation of Bulletproofs, SHA-256 serves as the default hashing mechanism, with the Barreto-Naehrig 256 (BN256) elliptic curve, reflecting standards similar to those used in Bitcoin. Bulletproofs operates under the zero-standard discrete logarithm hypothesis, achieving a substantial 128-bit security guarantee without the need for pre-initialized trusted settings. While there is a minor trade-off regarding increased verification time, Bulletproofs present notable advantages, including reduced proof size, rapid proof computation, and universality, making them particularly effective for range proofs that can integrate seamlessly with transactions.

Furthermore, Bulletproofs facilitate more efficient mass verification of arbitrary conclusions through zero-knowledge proofs. The protocol exhibits superior performance characteristics, whereby the bulk multiplier operates significantly faster than earlier proof versions leveraging full integration. The growth of proof size is logarithmic, with Bulletproofs allowing a doubling of range size complemented by a mere increase of 64 bytes in proof size, ensuring low byte counts. Overall, this prototype enhances the deployment of scalable and efficient identity management solutions leveraging blockchain technology.

Performance

Users of SEA chain apply on desktop computers running Windows with an Intel@Core (TM) i5-4950 CPU running at 4.40 GHz and 16.00 GB of RAM. The laptop used for the SEA chain system rollout is outfitted with an Intel@Core i7 3.30GHz processor, 16.00GB RAM, and a 2244MHz SSD, running macOS Mojave. Java 1.8 is used as the system development language.

PERFORMANCE COMPARISON

Performance	Scheme		
	WeIdentity	Sovrin	SeaChain
Credentials generation	0.042s	4.27s	0.032s
Proof generation	0.36s	1.2s	0.29s
Proof time	0.071s	0.006s	0.022s

Fig. 4: Performance Comparison

FUNCTIONALITY COMPARISON WITH OTHER SCHEMES

Functionality	Scheme			
	Uport	WeIdentity	Sovrin	SeaChain
Identity Privacy	✓	✓	✓	✓
Credential Privacy	×	×	✓	✓
Credential Nested Verification	×	×	×	✓
C-V Unlinkability	×	×	✓	✓
I-C Unlinkability	×	×	✓	✓
I-C-V Unlinkability	×	×	✓	✓

*C—Credentials, V—Verification, I—user social identity.

Fig. 5: Functionality comparison with other schemes

Our identity management system prototype is built on the Fisco Bcos blockchain, general statements proof systems, and Bulletproofs In this architecture, limited users are deployed as light nodes. A layered distributed proof system, a dual-credential model, and a distributed identity system that is distributed are the three primary parts of SEAChain.

To build cryptographic credentials, we use Pedersen commitment, the bulletproofs of range proof and non-malicious zero-knowledge, together with additional and OR proofs. In Bulletproofs, we employ SHA-256 is the default hash function, and Barreto-Naehrig 256 (BN256) is the default elliptic curve is used, which is identical to Bitcoin.

For the purpose of achieving attribute-level access control, SEA Chain constructs a systemic distributed proof system that is tree-based. Within the leaf nodes, there are numerous varieties of nested credentials that are included, whereas other nodes contain credential operators such as, respectively.

Performance of SEA Chain: The key steps of SEA Chain are the system setup, the generation of credentials, the verification of credentials, and the definition of algorithms. These processes are (Setup, Created ID, Create Credential, Create Claim, Ver Credential). Due to the fact that the Setup step is a one-time event, the performance analysis of Create DID, Create Claim, and the many different types of credential creation will be our primary focus, along with the verification of the CPU time and storage requirements that are specific to each of these procedures. The average amount of time required to create SEA Chain using the Created ID algorithm is roughly 0.4 seconds after 50 runs have been performed.

The algorithm known as Create Claim During the claim procedure, both plaintext claims and commitment claims are taken into consideration. A commitment claim is represented by the expression claim = att, com, U, whereas a plaintext claim is represented by the expression claim = att, val, U. When 15 characteristics are executed fifty times, the average amount of time required to generate a commitment claim is around four milliseconds, while the average amount of time required to construct a plain-text claim is approximately two milliseconds. As the number of qualities increases, the amount of time required to complete the task barely increases.

The Create Credential Algorithm is responsible for the generation of claims, credentials, proofs, and the uploading of proofs to the chain. These are the phases involved in the process of creating credentials. In the beginning, performance covers the Plaintext, Commitment, AND, and OR credentials. Because The Range credential has the most significant amount of time spent on its creation, we have decided to break it away. Figure 11 illustrates the amount of time required for the manufacturing and verification of the Range certificate. One of the most time-consuming credentials is the range credential, while the AND and OR credentials are rather complicated. The plaintext credential is the most efficient. This results in a linear rise in the amount of time required for construction. A blockchain with varying numbers of consensus nodes is depicted in Figure 6 as having variable timescales for the production of SEA Chain blocks and the implementation of transactions. The issuing of credentials and the up linking of credentials are both referred to as transactions within the SEA Chain system. In response to an increase in the number of consensus nodes, the amount of time it takes to generate a block exhibits a modest fluctuation, with a tendency toward a slight increase.

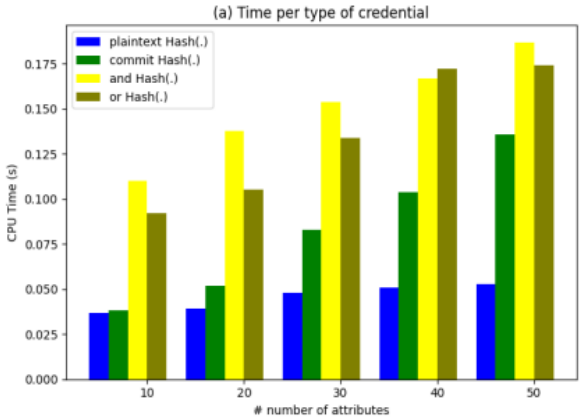

Fig. 6: Time per type of credential

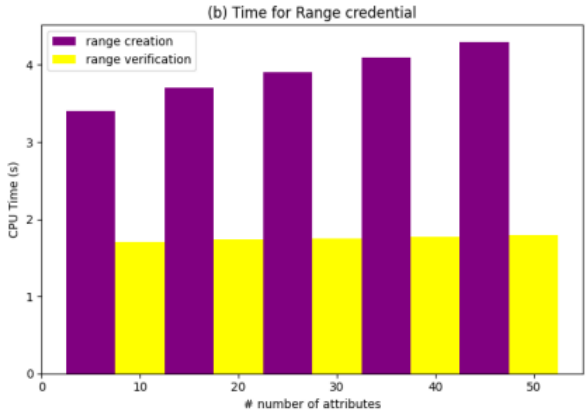

Fig. 7: Range credential time

The communication overhead that takes place between nodes and the PBFT consensus technique is what we are referring about! When it comes to communication, every node is connected to every other node in order to broadcast transactions and synchronize blocks. In regard to the latter, PBFT makes use of elections in order to prevent nodes from competing with one another for mathematical power in order to verify transactions on the test network. When conducting tests on a blockchain using a greater number of consensus nodes, the communication burden is increased, and the amount of time it takes to execute transactions is lengthened. The complexity of the message in PBFT is known as O(N2), where N represents the total number of nodes that make up the network. As a result of the network's transmission efficiency and latency, it is probable that bottlenecks will occur if the system has more than one hundred nodes. This could potentially limit the system's reliability. The Fisco Bcos blockchain that is part of SEAChain has the potential to manage a large number of consensus algorithms, one of which being RAFT, in order to fulfill the criteria of the system.

When the number of nodes increases, the amount of time that the block takes to commit to the modest oscillation that is contained within a predetermined range is the maximum amount of backend storage that can be achieved. This is the case regardless of the number of nodes. Now that the block and channel sizes have been defined and stabilized, it is possible for blocks to be committed to the backend storage without the need for time-consuming procedures like the consensus of the nodes.

In the usual situation of SEAChain, the identifier DID, the document containing the DID, the Plaintext credential, the Commitment credential, the Range credential, the OR credential and the AND credential, are all examples of identification credentials, there is a predetermined and regulated format for both the DID and the identifying documents. It is necessary to demonstrate the storage space that is assigned to each credential object is linear, and the bit size of the attribute data is similar to that.

A description of the VerCredential algorithm Each credential verification of a leaf node is comprised of three fundamental phases:

1) Confirming the legitimacy of the issuer's signature
2) Confirming the authenticity of identifiers
3) Reconstructing the credential digest (commit or hash) and comparing it with the blockchain proof

Each of these phases is essential to the process. In Figure 9, the performance of all types of credential verification is exhibited, and in Figure 7, the length of the Range proof's

credential building and verification process is discussed. There is a tendency for the amount of time required for the number of qualities increases, there will be a growth in the verification of credentials, even though verifying credentials is typically within the capability of the system. As we may have anticipated, the verification of credentials using plaintext is the quickest method since it does not necessitate proofs based on zero-knowledge. Due to the fact that the ubiquitous verification procedure is the same for all credentials, the amount of time that is necessary to verify each cryptographic credential the difficulty of the associated zero-knowledge verification approaches is the primary factor that determines the outcome.

A distributed proof system that is built on trees and is nested is being developed by SEAChain in order to address the problem of multiconditional verification of the verifier. This system enables proofs of plaintext credentials and cryptographic credentials, as well as fine-grained access control and mixed strategy proofs. A tree-based proof system is constructed when the verifier has completed the initialization of the verification conditions. All of the activities that were carried out on the credentials are reflected in the non-leaf nodes at this point, whereas the leaf nodes are a representation of the credentials themselves. The fact that the verification was successful is indicated by the fact that the entire tree returned a TRUE result. It should come as no surprise that the verification process in this particular instance is comparable to the search operation that is utilized in the tree-based proof system. Providing that if the proof system is successful, the result can be computed, it is possible to return the proof result without having to walk through the entire structure. For the purpose of assessing the effectiveness of the evidence system, we arranged and classified the numerous types of credentials and procedures. The ideal verification timings for Range proof, AND proof, OR proof, Commitment proof, and Plaintext are around O(logn), where n is the number of tree nodes. Plaintext verification is also an example of this. Plaintext verification is also the optimal time. There is a worst-case scenario in which the proof system has an O(n) proof time complexity. Furthermore, due to the fact that the system is capable of generating mixed strategy proofs, there is a temporal complexity that exists between the performance of Plaintext and Range for any combination of proof structures. As a consequence of this, the system's overall proof performance is productive. We enabled connection between these two personal computers by establishing a local network on each of them. The average latency of the network is 14 milliseconds, while the download and upload speeds are 38.8 megabits per second and 12.21 megabits per second, respectively. The AND, OR, Plaintext, and Commitment credentials, as well as the technique for credential construction, Due to the fact that the Range credential has the most amount of time spent on its creation, we separated it. Figure 7 serves as an illustration of the process of creating and verifying the Range certificate. Plaintext credentials are the most efficient kind of credentials in terms of construction time, while and, or, and range credentials are the most time-consuming types of credentials. This results in a linear rise in the amount of time required for construction. Figure 6 illustrates the amount of time required for the production of blocks and the commit of transactions for SEAchain in blockchains that have varying numbers of consensus nodes. The SEAchain system functions as a mechanism for issuing and uplinking credentials, treating these actions as transactions. The time required to create a block within this system demonstrates modest volatility, exhibiting a gradual increase as the number of consensus nodes in the network rises. Despite the growing number of nodes, the time for a block to be committed to backend storage remains consistent within a narrowly defined variance. This efficiency allows for block commitment without necessitating lengthy procedures for node consensus, owing to the established and reliable definitions of both block size and channel size. In addition to the AND, OR, Range, and Plaintext credentials, the DID document storage space in SEAchain is also included.

Commitment certification encompasses fifteen criteria in the average state. Figures 8-10 provides a description of the technique for constructing and verifying credentials for the Range proof. In general, the amount of time required for credential verification is found to be within the capabilities of the system, despite the fact that it has a tendency to increase with the number of attributes. As was to be expected, the verification of credentials using plaintext is the quickest method since it does not participate in proofs that need zero knowledge. The length of time it takes to validate a particular cryptographic credential is ultimately determined by the difficulty of the appropriate zero-knowledge verification procedures. This is due to the fact that the common verification process is the same for all credentials. The optimal verification time for each of the following proof types is about O(logn), where n is the number of tree nodes. These proof types are AND, OR, Commitment, Range, and Plaintext (Table 1).

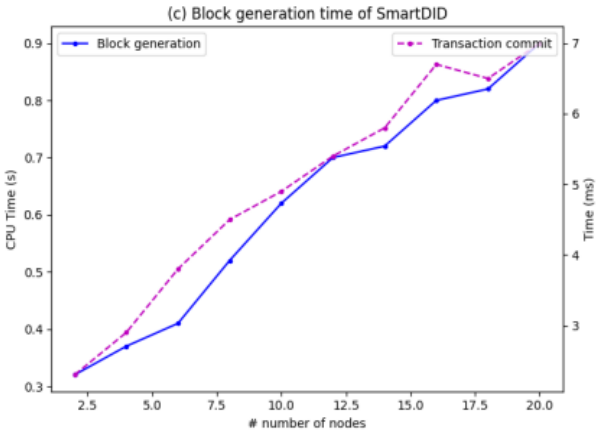

Fig. 8: SEAchain's block generation time

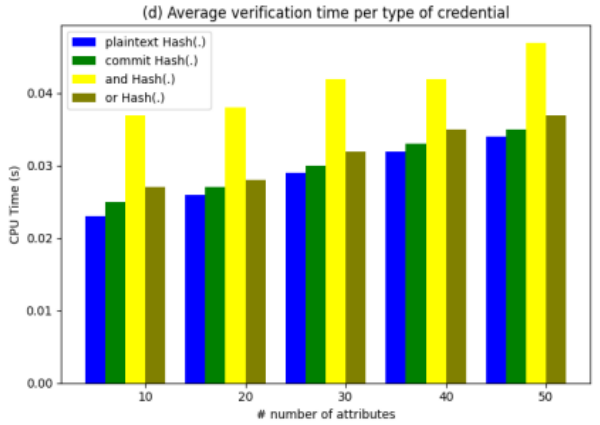

Fig. 9: Credential verification time

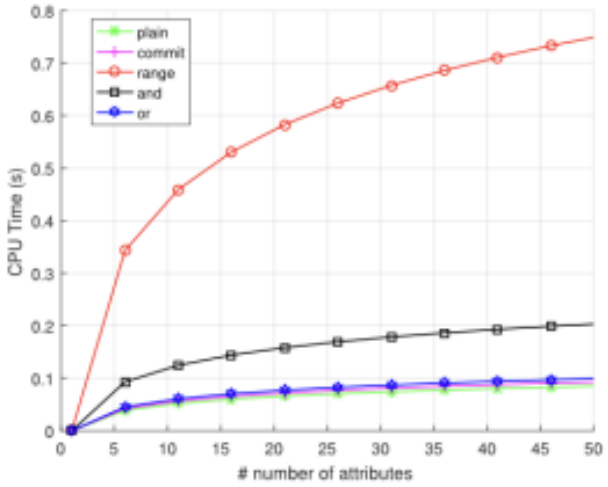

Fig. 10: Credential verification time

Table 1: Notations

Symbol	Description
ID	The distinct master identity for oversight
PWD	The anonymous identities
W	A q-order cyclic group
r, p	Two arbitrary w generators
u	The integer vector with n dimensions, $u = \{u_1,...,u_n\}$
D	The generators of D, $D = \{D_1,...,D_n\}$ in n dimensions
(Ek, Dk)	The pair of public and private keys, $Ek = B$, Dk
Hash(.)	Credential
$Enc_c(.)$	Function of commitment
Enc	Function of commitment value
H(.)	A declaration using the attribute-value-user format
PS	The user's attribute vector
TA	The vector of values to attribute
V	Identity user
xy	An accessible public token
U	The user V's signature proof of claim
ε	
M	The quantity of nodes in the committee
O_j	The committee nodes in j-th
εj	O_j's proof of claim
ε	The sum of all the signatures ε_j
εI	The credential proof signed by the Issuer
B	A hash function
g_q	The credential's commitment

The proof time complexity of the proof system is equivalent to O(n) in what is considered to be the worst possible circumstance. For every combination of proof structures, there is a temporal complexity between the performance of Range and Plaintext. This is because the system allows for mixed strategy proofs, which means that the system allows for mixed strategy proofs. As a consequence of this, the system's overall proof performance is productive. Figures 4 and 5 illustrate the comparison of the performance and functionality of currently available schemes. Based on the results shown in Figure 4, the intended plan has been successful. In comparison to other schemes, it enables a greater number of security and privacy features, as shown in Figure 5. The proposed plan is therefore more effective and efficient than the alternative.

Conclusion

SEAchain is a distributed identity management system that includes an identity system as its core component. We developed a distributed identity system that is supervisable, unlinkable, and resistant to Sybil attacks. To be more precise, we used several user IDs to break the connection between various credentials and a single master ID to monitor and stop Sybil attempts. Moreover,

we developed a dual-credential architecture that has plaintext credentials as well as privacy credentials that are based on commitment and bulletproofs in order to safeguard the confidentiality of sensitive attributes and data that is stored on the blockchain. In addition, we built a methodical distributed proof system to address more complicated issues and solve broad assertions by combining the fundamental credential proofs to demonstrate expertise.

Acknowledgment

Thank you to the publisher for their support in the publication of this research article. We are grateful for the resources and platform provided by the publisher, which have enabled us to share our findings with a wider audience. We appreciate the efforts of the editorial team in reviewing and editing our work, and we are thankful for the opportunity to contribute to the field of research through this publication.

Funding Information

The authors have not received any financial support or funding to report.

Authors Contributions

S. Balachander: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing-original draft preparation.

A. Murugan: Supervision and project administration.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

Conflicts of Interest

The authors declare no conflict of interest.

References

Ahmed, E., & Kim, H. (2023). Blockchain-enabled secure data sharing with attribute-based encryption. *IEEE Transactions on Services Computing*, *16*(1), 49–62.

Alharbi, M., & Zohdy, M. (2023). Privacypreserving identity and credential management using blockchain. *Computers and Electrical Engineering*, *106*, 108572.

Ali, A., & Hussain, S. (2024). A comprehensive survey on blockchain-based digital identity systems. *IEEE Access*, *12*, 77452–77470.

Almeida, J., & Costa, D. (2024). Toward secure decentralized identities in smart environments. *IEEE Internet of Things Journal*, *11*(4), 5521–5536.

Bai, G., & Xu, T. (2024). Privacy-preserving blockchain authentication using hybrid encryption. *Information Sciences*, *650*, 119676.

Chen, X., & Zhang, L. (2024). Zero-knowledge proof-enabled blockchain identity ecosystem. *Computers and Electrical Engineering*, *115*, 109197.

Ferrari, E., & Cinque, M. (2023). A systematic review on self-sovereign identity: Concepts, technologies, and applications. *ACM Computing Surveys*, *55*(11), 1–36.

Guo, R., Shi, H., & Zhao, Q. (2022). Identity privacy preservation framework based on blockchain and zeroknowledge proofs. *Journal of Network and Computer Applications*, *200*, 103327.

Hsu, C.-H., & Chen, Y.-C. (2024). Towards decentralized self-sovereign identity systems using verifiable credentials. *IEEE Access*, *12*, 18245–18260.

Huang, X., Xu, J., & Liu, Q. (2022). A secure and privacy-preserving identity authentication scheme using blockchain. *Information Sciences*, *600*, 209–224.

Jia, W., & Liu, C. (2024). Design of a blockchainbased privacy-preserving identity protocol.*IEEE Transactions on Industrial Informatics*, *20*(7), 9552–9563.

Jiang, X., & Han, J. (2024). Decentralized identity verification with dual credentials on blockchain. *Future Generation Computer Systems*, *155*, 112–125.

Kang, J., Yu, F. R., & Kim, D. (2023). Blockchainbased secure and privacy-aware authentication in edge networks. *IEEE Transactions on Wireless Communications*, *22*(6), 3765–3778.

Khan, M., & Malik, H. (2024). A privacy-enhanced blockchain model for secure identity and transactions. *IEEE Access*, *12*, 50211–50224.

Kou, W., & Li, Y. (2024). Blockchain-based selective disclosure identity framework. *IEEE Transactions on Cloud Computing*, *12*(1), 401–412.

Kumar, S., & Zhang, P. (2022). A lightweight blockchain identity framework for IoT environments. *IEEE Sensors Journal*, *22*(14), 13821–13830.

Li, W., Chen, M., & Wang, L. (2021). A privacypreserving decentralized identity management system based on blockchain. *IEEE Internet of Things Journal*, *8*(15), 12014–12027.

Liu, J., Huang, C., & Yu, F. R. (2022). Blockchainbased trust management in decentralized systems: A comprehensive review. *IEEE Communications Surveys & Tutorials*, *24*(3), 1626–1657.

Lopez, D., & Perez, J. (2024). A survey on privacy-enhancing blockchain technologies for identity. *Computer Communications*, *224*, 105–124.

Nguyen, T., & Hu, J. (2024). A privacy-preserving credential verification framework with blockchain and ZKPs. *Sensors*, *24*(3), 1055.

Patel, R., & Sharma, V. (2024). Secure credential linking and verification on blockchain. *Applied Soft Computing*, *147*, 110803.

Peng, Y., Zhang, L., & Chen, K. (2022). Decentralized identity authentication with zk-SNARKs on blockchain. *IEEE Access*, *10*, 81454–81466.

Pillai, S., & George, A. (2024). Blockchain assisted zero-knowledge credential management system. *Journal of Parallel and Distributed Computing*, *188*, 104950.

Qin, Z., & Li, S. (2024). Blockchain-driven confidential identity management architecture. *Security and Communication Networks*, *2024*, 1–14.

Rahman, Md., & Hossain, S. (2024). A verifiable credential-based blockchain identity architecture. *Journal of Systems Architecture*, *148*, 103129.

Rasheed, I., & Salah, K. (2024). Blockchain for privacy-preserving data sharing and identity. *IEEE Communications Surveys and Tutorials*, *26*(1), 250–280.

Ryu, J., & Lee, H. (2024). A dual-credential decentralized identity model for regulated environments. *Computers and Security*, *130*, 103351.

Sharifi, M., Dehghantanha, A., & Parizi, R. M. (2021). Self-sovereign identity on blockchain: Trends and challenges. *Future Generation Computer Systems*, *125*, 887–902.

Singh, A., & Gupta, P. (2024). Blockchain-based distributed identity with privacy guarantees. *Journal of Information Security and Applications*, *82*, 103700.

Sun, X., Wang, Y., & Luo, E. (2022). A secure verifiable digital identity model using blockchain. *Computers and Security*, *120*, 102795.

Wang, H., Qin, Z., & Ren, K. (2021). Blockchain based data privacy management with fine-grained control. *IEEE Transactions on Dependable and Secure Computing*, *18*(5), 2438–2452.

Wang, X., & Lin, J. (2024). Secure blockchain identity architecture with unlinkability guarantees. *IEEE Transactions on Network and Service Management*, *21*(2), 145–159.

Yang, F., Li, J., & Zhang, X. (2023). Secure blockchain identity verification based on dual-credential cryptography. *IEEE Transactions on Information Forensics and Security*, *18*, 2031–2045.

Yuan, S., & Tang, W. (2024). zk-SNARKs powered private authentication scheme on blockchain. *Symmetry*, *16*(5), 622.

Zhang, Y., Chen, X., & Li, J. (2021). Blockchainbased privacy-preserving identity management: A survey. *IEEE Access*, *9*, 134491–134512.

Zhao, H., Wang, H., & Yu, S. (2023). Blockchainbased privacy-preserving data access control for distributed systems. *Future Generation Computer Systems*, *141*, 167–180.

Zhou, L., Diro, A., Saini, A., Kaisar, S., & Hiep, P. C. (2024). Leveraging zero knowledge proofs for blockchain-based identity sharing: A survey of advancements, challenges and opportunities. *Journal of Information Security and Applications*, *80*, 103678. https://doi.org/10.1016/j.jisa.2023.103678

Zyskind, G., Nathan, O., & Pentland, A. (2015). Decentralizing Privacy: Using Blockchain to Protect Personal Data. In 2015 IEEE Security and Privacy Workshops (pp. 180-184). IEEE. https://doi.org/10.1109/spw.2015.27