IMPROVING THE SECURITY AND PRIVACY OF DECENTRALIZED IDENTITY MANAGEMENT IN MULTI-CONTROLLER SCENARIOS

HUIJIONG YANG

MPhil

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University

Department of Computing

Improving the Security and Privacy of Decentralized Identity Management in Multi-Controller Scenarios

Huijiong Yang

A thesis submitted in partial fulfillment of the requirements for

the degree of Master of Philosophy

May 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Huijiong Yang

Abstract

Decentralized identity (DID) is pivotal to Web3 applications as it empowers users to manage their identities and credentials without relying on any central authority, thereby enhancing privacy, security, and user autonomy. Existing research on DID primarily focuses on single-controller scenarios, where controllers have complete privileges for identity and credential management. The multi-controller scenario is also an important and indispensable component outlined by the W3C DID standards, while its privacy and security issues have not yet been fully explored. These issues stem from both existing coarse-grained identity management and the intrinsic characteristics of blockchain systems, such as data transparency and high ledger-commit latency. In this thesis, we aim to solve these problems and construct privacy-preserving and secure identity management schemes for DID in multi-control scenarios. We carry out the following work.

In our first work, to solve the problems caused by coarse-grained identity management adopted current schemes, i.e., identity impersonation by malicious controllers and high key recovery overhead, we propose MoDID, a fine-grained identity management scheme for multiple controllers, which complies with the DID standard proposed by W3C. Our solution allows multiple controllers to control DID subjects flexibly and reliably through hierarchical controller management. Additionally, we design a secure and low-overhead key recovery scheme to reduce the risk of identity loss. The controllers only rely on the social control recovery in case other controllers cannot execute

replacing operations. Finally, we implement MoDID on the Sepolia Ethereum Test Network to evaluate the effectiveness of our proposed scheme. The result demonstrates that our system allows multiple controllers to manage a single identity with lower gas consumption and time consumption than the state-of-the-art.

In our second work, we find two new attacks in multiple controller scenarios caused by the intrinsic characteristics of blockchain systems. We also propose a privacy-preserving and secure identity management scheme to defend against them. The first proposed controller-correlation attack allows an attacker to infer relationships between different subjects by correlating the public keys uploaded by controllers in the blockchain. To avoid this kind of privacy leakage, we propose a masking scheme based on the Merkle tree, which allows the controllers to prove their ownership over the multi-controller identities without publicizing the plaintext of their public keys. The other identity impersonation attack exploits insecure controller revocation caused by high block synchronization latency. To resist this attack, we propose a lightweight authentication scheme where the holders provide digest freshness proof and the verifiers only need to download block headers. Finally, to evaluate the feasibility of our proposed scheme, we implement our system on the Sepolia TestNet. The experimental result demonstrates that our system can prevent these attacks with acceptable gas consumption and time consumption, compared with the state-of-the-art.

Publications Arising from the Thesis

1. Huijiong Yang, Rui Song, Bing Chen, Yubo Song, and Bin Xiao "MoDID: Decentralized Identity Management for Multiple Owners", accepted by *IEEE International Conference on Communications (ICC)*), Denver, CO, USA, 9 - 13 June 2024.

2. Huijiong Yang, Bin Xie, Jianhuan Wang, Guyue Li, and Bin Xiao "Privacy-Preserving and Secure Decentralized Identity Management for Multiple Controllers", accepted by *IEEE Global Communications Conference (GLOBECOM-2024)*, Cape Town, South Africa, 8–12 Dec. 2024.

Acknowledgments

Firstly, I am deeply grateful to my supervisor Prof. Bin Xiao. Prof. Xiao gave me invaluable guidance and support throughout my thesis-writing journey. Prof. Xiao provided me with guiding suggestions on the research direction. He also offered me careful guidance on the problems I met during the thesis writing process.

Secondly, I would also like to thank my colleagues, Mr. Rui Song, Mr. Bin Xie, Mr. Jianhuan Wang, Dr. Guyue Li, Dr. Zecheng Li, Mr. Zheng Tianyu, and Mr. Huang Chengpeng. They have offered me great support for both my research and life in Hong Kong. It has been my privilege and honor to collaborate with such exceptional colleagues and friends.

Finally, I express my profound gratitude to my beloved family for their unwavering love, unwavering support, and boundless confidence in me throughout the two years. I am also grateful to my other friends who have provided generous support and invaluable advice over the past two years. Their encouragement has been instrumental in my academic and personal growth.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Decentralized Identity

The development of decentralized ledgers [26, 61, 77, 13] has given the birth of Web3, which aims to establish an open and decentralized ecosystem. Identity management plays a vital role in Web3 applications for authentication and access control. The ForgeRock 2023 Identity Breach Report [1] reveals that approximately 1.5 billion identities have been compromised in nearly a year, resulting in incurring an average cost of nearly $9.4 million per breach. Decentralized identity (DID) is based on decentralized ledgers, which is a promising solution for Web3. DID enables the holders to autonomously control their identity and credentials, reducing security risks associated with data centralization in traditional authentication schemes. Especially, the issuers and holders upload their public keys to the decentralized ledgers. These public keys are used by the verifiers to verify credentials. Generally, DID can rely on any decentralized ledgers, like the InterPlanetary File System (IPFS) or blockchain system. However, blockchain systems use hash links for immutability features compared to

[1] https://www.pingidentity.com/en/resources/content-library/analyst-reports/3763-2023-forgerock-identity-breach-report.html

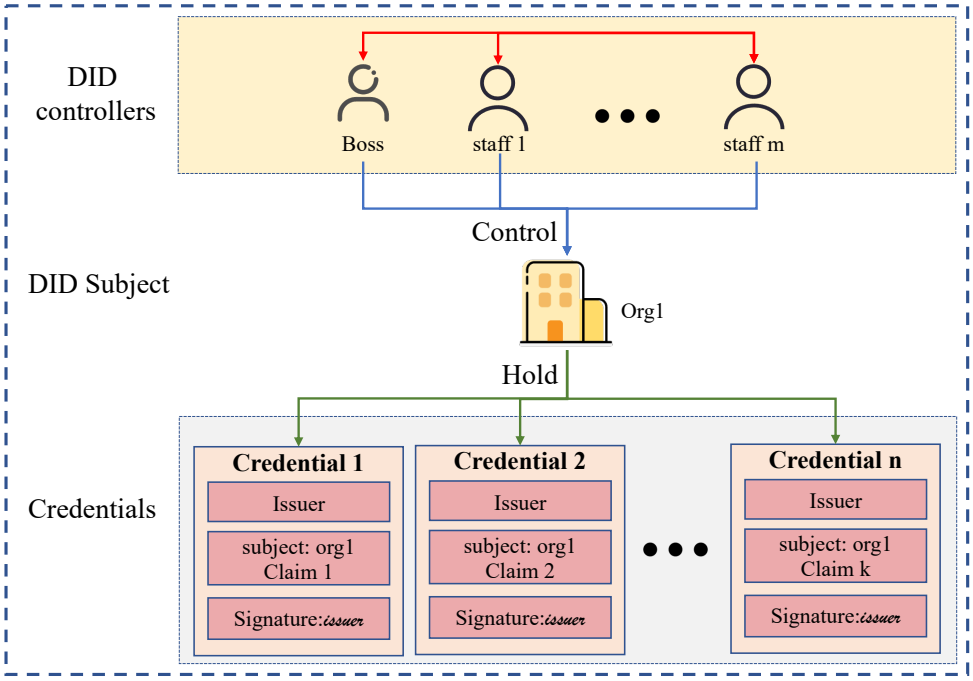


Figure 1.1: A Generic Architecture of a DID Subject with Multiple Controllers and Credentials.

other decentralized ledgers. Blockchain enables the verifiers to get the correct public keys for verification. Thus, many DID systems introduce blockchain.

In a DID system, holders obtain credentials from issuers and selectively disclose them to a verifier for identification or authentication [50]. Fig. 1.1 illustrates the structure of a typical DID system in which the subject can be a person, physical entity, or organization. The credential portrays the attributes of the DID subject, and its controller may be the subject itself or multiple other entities. The controller manages the subject with its private key and thus enjoys full sovereignty over credentials. Thus, it is vital to make controllers use and manage credentials in a secure and reasonable way.

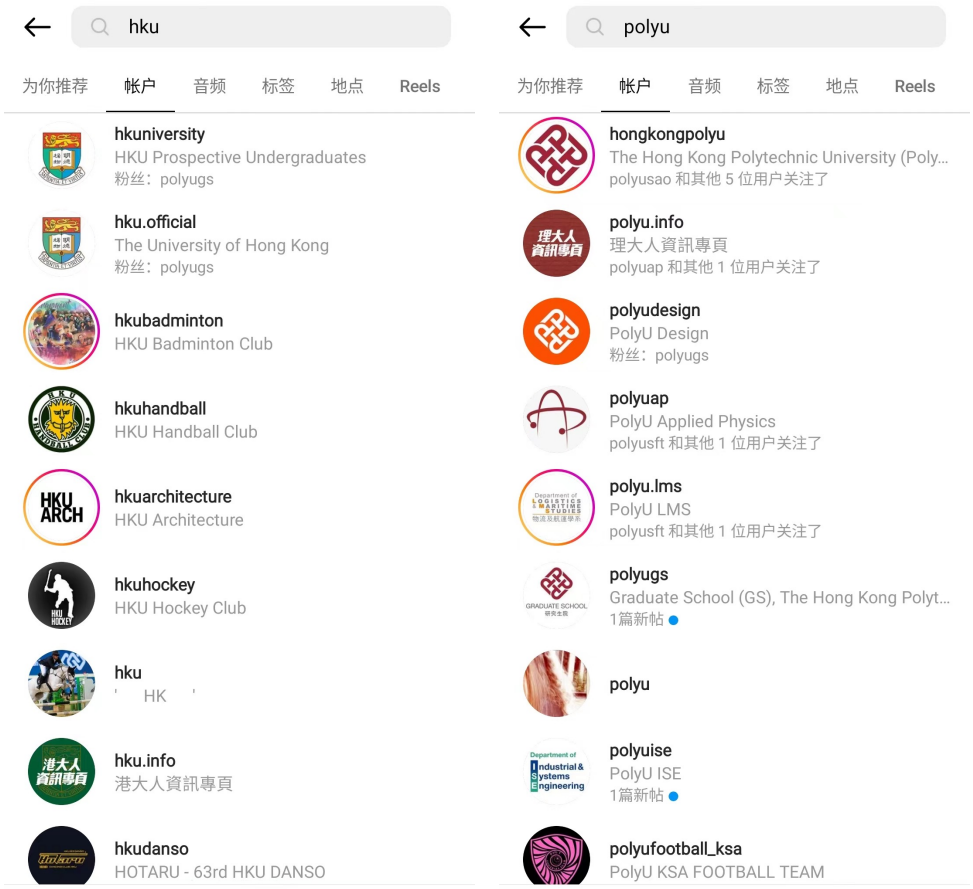


Figure 1.2: Organization Accounts in Instagram.

1.2 Motivation

Most of the existing DID schemes focus on a single-controller structure [33, 40, 15, 27] where a subject is fully controlled by one controller. However, as elucidated in the DID standard proposed by the World Wide Web Consortium (W3C), many application scenarios require joint control of a subject by multiple controllers. For example, an organization may require multiple managers to jointly manage an organization's credentials. For example, as shown in Fig. 1.2, there are many accounts of organizations in the Instagram application [2], such as hkuniversity and hongkongpolyu. Identity management in multi-controller scenarios has a wide range of applications,

[2]https://www.instagram.com/

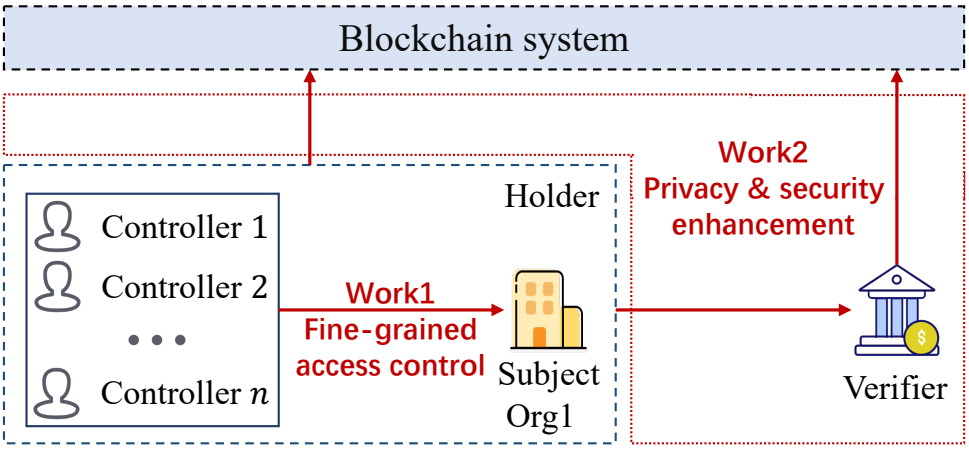


Figure 1.3: Problem Overview.

but little work focus on this area. The uPort system [31] allows multiple controllers to jointly control an identity. However, the uPort system only grants the same privileges to all controllers. This can lead to serious security and privacy problems, impeding the application of DID in multi-controller scenarios.

Even though the multi-controller scenarios have wide promising applications, there are many serious security and privacy issues encountered by the existing DID systems. These issues stem from both coarse-grained identity management and the intrinsic characteristics of blockchain systems, such as data transparency [8] and high ledger-commit latency [60]. Coarse-grained identity management schemes can result in security vulnerabilities, including identity impersonation attacks by malicious controllers during the management process and unnecessary key recovery overhead due to inflexible schemes. The inherent characteristics of the blockchain system can lead to security and privacy issues, such as identity impersonation attacks stemming from insecure controller revocation in asynchronous environments and controller-correlation attacks even from external adversaries. These issues pose a great threat to identity security and privacy in multi-controller scenarios. Therefore, it is urgent to solve these problems, and enable controllers to manage multi-controller identities in a secure and privacy-preserving manner. In our first work, we concentrate on addressing security

issues arising from coarse-grained identity management, while our second work aims to tackle challenges arising from blockchain characteristics.

1.3 Thesis Contribution

The objective of the thesis is to design a W3C-standard compatible, secure, and privacy-preserving identity management scheme for multiple controllers. This system aims to address the security and privacy issues arising from both coarse-grained access control mechanisms and the unique characteristics of blockchain technology. We summarize our work in Fig. 1.3. We present the following contributions to achieve this goal.

- In our first work, we focus on the security issues caused by coarse-grained access control, i.e., identity impersonation by malicious controllers and high key recovery overhead problems. To solve the problems, we propose MoDID, which is a novel fine-grained identity management scheme for multiple controllers by hierarchical controller management and modifying the key recovery strategy. 1) To resist identity impersonation attacks, we propose a hierarchical controller management scheme considering the inherent access control in the organization scenarios. The controllers are categorized into three distinct roles based on their reliability. 2) To reduce the key recovery overhead, we devise an optimized and flexible key recovery scheme and reduce the risk of identity loss. Specifically, the controllers rely on the relatively high-overhead social key recovery strategy, only in instances where controllers with corresponding permission cannot execute controller management operations.

- In our second work, we find two new attacks caused by blockchain characteristics, i.e., controller-correlation attacks and impersonation attacks by revoked controllers. We also propose a low-overhead scheme to defend against both

attacks. 1) To resist to controller-correlation attacks, we design a privacy-preserving controller management scheme based on the Merkle tree. We allow the controllers to provide ownership proof without publicizing their public keys. 2) To solve the insecure controller revocation problem, in our scheme, the holders provide digest freshness proof and the verifiers only need to download block headers, rather than synchronizing heavy blocks. The verifiers can access up-to-date public keys efficiently.

In summary, we design a W3C-standard compatible identity management scheme for multiple controllers characterized by secure and privacy-preserving features. We address the security and privacy issues caused by both coarse-grained access control and blockchain characteristics. Our work promotes the application of DID in multi-controller scenarios.

1.4 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 introduces the main technologies we used in this thesis. Chapter 3 presents our hierarchical identity management scheme for identity impersonation and high-overhead key recovery problems. Chapter 4 presents our secure and privacy-preserving identity management scheme for controller-correlation attacks and insecure controller revocation problems. Finally, we conclude the thesis and give our suggestions for future research in Chapter 5.

Chapter 2

Background

2.1 The Era of Web 3.0

2.1.1 The Overview of Web 1.0 and Web 2.0

Web 1.0 [54] refers to the early days of the World Wide Web, roughly from the early 1990s to the early 2000s. It is known as the static web. Few content creators can generate figures and text information on the website. The vast majority of users are limited to passively viewing content. They simply act as consumers of content and communicate with the website in a one-way manner, with limited user interaction and dynamic content.

Web 2.0 [19] emerged around the mid-2000s. It represents a significant shift in how the web is used, focusing on user-generated content, social interaction, and dynamic, interactive websites. A Web 2.0 website allows users to interact and collaborate with each other through social media dialogue as creators of user-generated content in a virtual community. The users can read and create content on the website, as well as upload or download files stored in the database. It also catalyzes the emergence of applications with user interoperability, like social media platforms and content-sharing

sites. However, in the Web 2.0 era, users rely on large Internet enterprises to provide services. The centralized private data management scheme may potentially lead to serious privacy leakage problems. These enterprises may be attacked by hackers or sell users' privacy for the sake of profit. For example, in 2021, Facebook was attacked due to a code vulnerability. As a result, the privacy data of more than 533 million users is exposed, including their phone numbers, full names, locations, as well as birthdates [21].

2.1.2 Blockchain Technology

A blockchain system is one of the distributed ledgers based on the peer-to-peer network. It contains increasing blocks with related data with immutable and traceable characteristics. These blocks are linked together through the cryptographic hash functions. Specifically, the full nodes jointly keep these blocks and pack new blocks into the blockchain system through Byzantine fault-tolerant consensus protocol[5, 65].

Bitcoin [41] is proposed by an individual or group known as Satoshi Nakamoto in 2008. It is the first cryptocurrency. The Bitcoin system uses blocks to store the transaction data. These blocks are linked using cryptographic hash functions to provide the tamper-proof. Specifically, the blocks are made up of two parts, i.e., block header and block body. Block body contains transactions using Merkle tree algorithms. With the block headers as commit, it is easy for the light nodes to verify whether a transaction is contained by a block. The miners run the Proof of Work (PoW) protocol [17] to add new blocks to the blockchain.

Ethereum [6], introduced by Vitalik Buterin in 2015, is a decentralized blockchain platform. Central to Ethereum's innovation is the concept of smart contracts [67, 76], programmable self-executing agreements that enable automated transactions without the need for intermediaries. This platform facilitates a wide array of decentralized applications [7, 9], like supply chain management and voting systems, thus significantly

expanding the potential use cases of blockchain technology. Additionally, Ethereum uses the proof-of-stake (PoS) algorithm to get consensus. The average block generation time on Ethereum is 15 seconds, which significantly improves the scalability of the Ethereum system.

Apart from the two main blockchain systems, there are several other notable blockchain platforms. For example, Ripple [2] is a blockchain platform primarily focused on providing fast and low-cost cross-border payment solutions for financial institutions. Solana [70] designs an innovative scheme to solve the issues related to scalability and high transaction costs. To improve the scalability of existing blockchain systems, Polygon [48] is a Layer 2 scaling solution designed to enhance the scalability and transaction speed, while reducing transaction costs. It aims to provide developers and users with a faster and more cost-effective transaction experience. The development of blockchain systems makes it possible to build a web ecosystem with decentralized characteristics.

2.1.3 The Overview of Web3

The centralized architecture in Web 2.0 raises concerns about information centralization and data leakage. The development of blockchain technologies [57] has catalyzed the emergence of Web3 [30, 25]. Web3 aspires to establish an open and decentralized ecosystem, which is often described as the next phase of the internet [10]. Firstly, It aims to provide distributed Internet services without reliance on trusted third parties. It reduces the risk of privacy leakage problems caused by data centralization. Another important characteristic is user-centralized [49]. This means that by using cryptography algorithms, the users can have full control over their data, including identities, credentials, as well as tokens. The users can decide when to disclose their own information to others. It greatly reduces the risk of user privacy leakage.

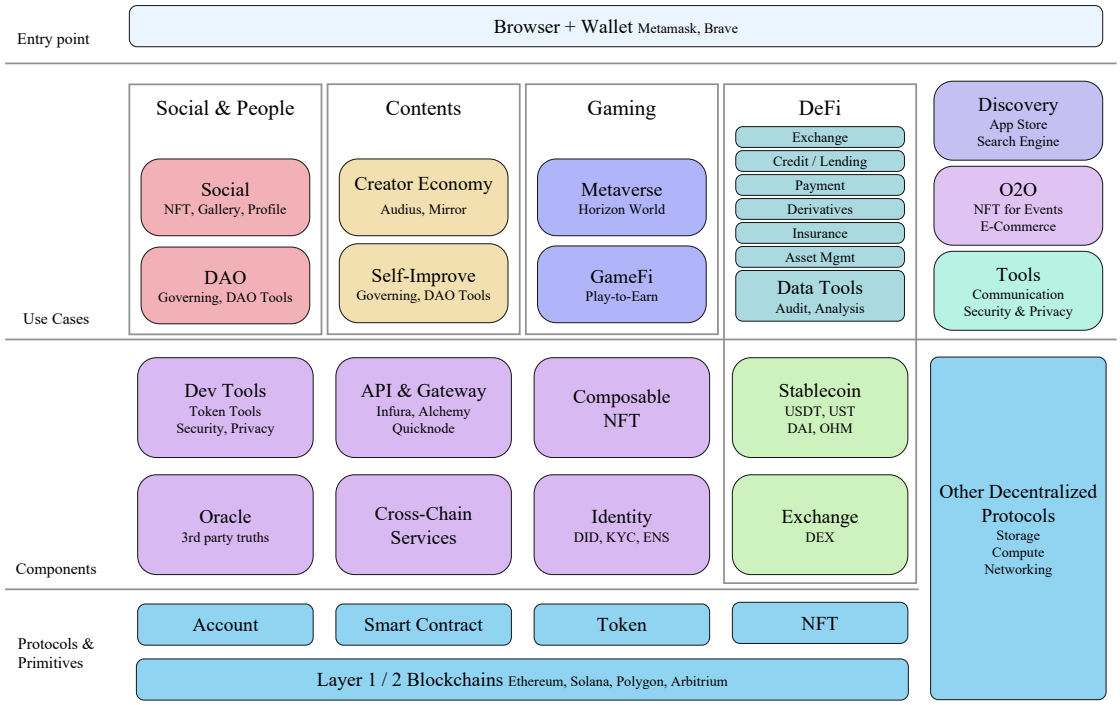


Figure 2.1: The Component of Web3.

Fig. 2.1 plots the elements and their relations in a Web3 ecosystem [1]. There are four layers in the ecosystems, i.e., protocols and primitives, components, use cases, and entry points. In the protocols and primitives layer, the Web3 ecosystem needs blockchain systems with good scalability [56], accountability [42], and privacy-preserving characteristics, due to the existence of massive users. Based on the blockchain system, there are many basic components. For example, Oracle [32] makes it possible to extract external data securely to the blockchain. Cross-chain services [43] allow users to interact efficiently in different blockchain systems, like transferring transactions. Identity management plays an important role in the Web3 ecosystem, since it has a direct impact on the users' privacy and digital assets. In a decentralized ecosystem without reliance on centralized services, it is challenging to allow users to control their identities and credentials in a secure and privacy-preserving manner. There are many challenges in the process of identity management, including Sybil attacks, the

[1] https://eth.mirror.xyz/eQwtbeWM8Xq37oxf0nlSxwkFpG33qoxPa8o_wb7YbbY

linkability of identifiers, and identity loss problems. Given the basic components, the developers can develop several decentralized applications for different scenarios, such as Decentralized Autonomous Organization (DAO) [66], decentralized games, and Decentralized Finance (DeFi) [75]. As for the entry point, the users rely on wallets [52] to use the decentralized applications. The wallets are used to securely keep the users' private keys, as well as their related tokens and credentials. The wallets can be hosted wallets, non-custodial wallets, and hardware wallets. Through the wallet, the users can interact with the blockchain systems or other corresponding decentralized databases [4].

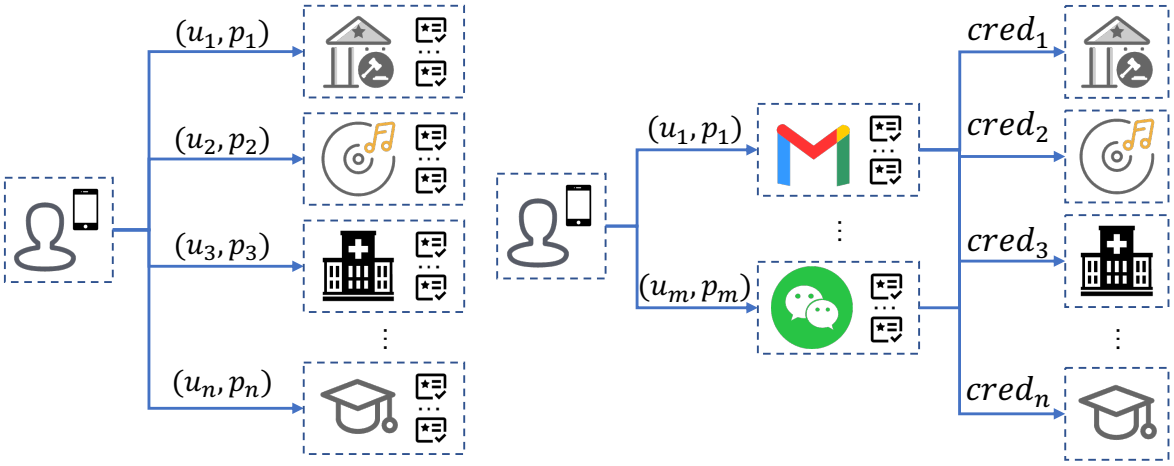
2.2 Decentralized Identity Systems

Identity management is critical for authentication and access control in Web applications. In the Web2 era, users rely on centralized [16] and federated identity [58] management structures for identification and authentication. Web3 [47] aims to achieve an open ecosystem with decentralized characteristics. The centralized nature and high risk of privacy leakage make these schemes difficult to apply in the Web3 era. DID, based on decentralized ledgers, serves as a robust and privacy-preserving identity infrastructure. It is a promising authentication scheme in the Web3 era.

2.2.1 Traditional Identity Management Solutions

The centralized identity and federated identity management schemes play an important role in the Web 2.0 era.

In a centralized identity system [46], the users rely on a single centralized web service to control and manage their identity information and credentials, as shown in Fig. 2.2 (a). Specifically, the individuals send their personal information to the centralized web services. After verification, the web services keep the user's credentials.



(a) Centralized Identity Management Scheme. (b) Federated Identity Management Scheme.

Figure 2.2: Traditional Identity Management Scheme.

However, this system raises concerns about privacy and security. For users, they need to remember multiple usernames and passwords for authentication. While for web services, different web services isolatedly keep the credentials, which will lead to the waste of storage resources. Besides, a proliferation of credentials and users' enhanced demand for privacy protection place a great burden on web services. These web services can disclose users' private data or credentials for-profit purposes or due to attacks. For example, Didi Global Inc. illegally collected nearly 12 million screenshots and 107 million pieces of passengers' facial recognition data, and more than 167 million records of location data in 2022 [62].

Federated identity management scheme [58, 23] allows users to use the same set of credentials for authentication and access control across services [11], as shown in Fig. 2.2 (b). Specifically, federated identity allows different web services to appoint one authority as an identity provider (IdP), such as Wechat. IdP can keep users' authentication information, such as credentials or biometric attributes. Users can use the credentials verified by IdP to access different web services without needing

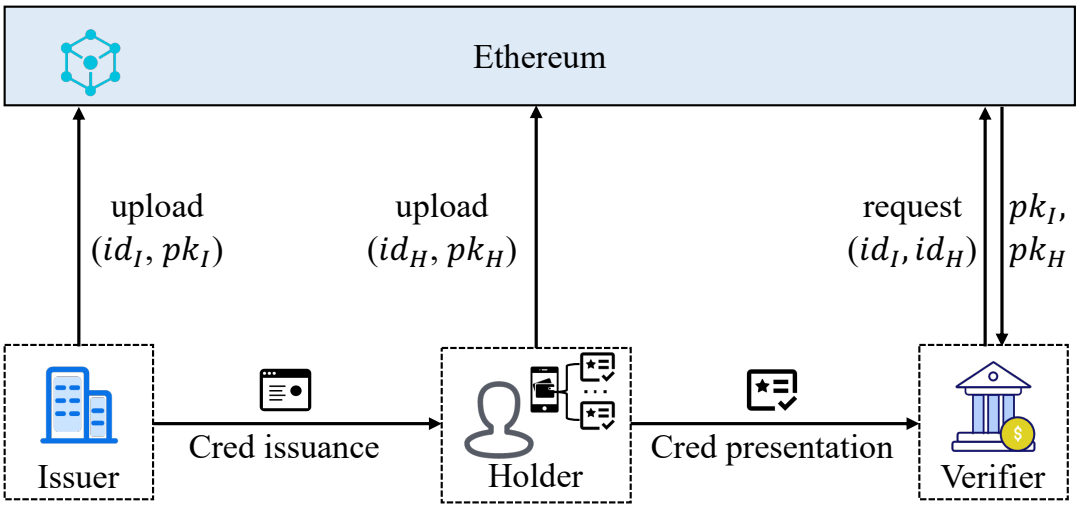


Figure 2.3: Decentralized Identity Management Scheme.

to register or log in separately to different systems. Federated identity allows web services to share authentication information. It also enables the users to use one identity to access resources in different web services. However, in such schemes, the trust relationship leaves out the individual, giving rise to privacy and data protection concerns.

2.2.2 Decentralized Identifier and Verifiable Credential

Blockchain technology [41], and decentralized ledgers have given rise to Web3, which is aimed at achieving an open ecosystem with decentralized characteristics. The centralization nature and the risk of privacy leakage make centralized and federated identity management difficult to apply in the Web3 era. DID [3] is proposed to rectify the shortcomings of centralized identity management structures. DID system serves as a robust and privacy-preserving identity infrastructure. DID enables the users to autonomously control their identity and credentials, reducing security risks associated with data centralization in traditional authentication schemes.

DID scheme allows users to have full control over their identities and credentials.

It has promising applications in the Web3 ecosystem. Since DID was proposed, its security and privacy issues have attracted attention from researchers. In 2022, W3C proposes the standards of DID [50, 63]. These standards clarify the participating entities, issues, holders, and verifiers, as shown in Fig. 2.3. They also clarify the authentication process and specify the credential format. According to the standards, the holders can obtain credentials from the issuers and autonomously present them to verifiers without the involvement of the issuers. Specifically, the issuer is responsible for creating and issuing verifiable credentials to the holders. Issuers are usually authorities, like governments or universities. The issuer verifies the individual's identity information and converts it into digital credentials or encrypted identity information for secure use on the network. The holders are usually web users, who want to use their credentials for identification and authentication. The holders can collect credentials from issuers and store these credentials in personal devices or digital wallets. The holders can autonomously present credentials to verifiers. In the presentation process, the holders can decide when, with whom, and which attributes to disclose. The verifiers can be service providers, employers, and websites. They participate in the authentication process by verifying the credentials. Different from centralized identity and federated identity schemes, the credential verification process does not involve the participation of issuers. After receiving the credential presentation, the verifiers only fetch the public key recorded in the block and then verify the validation of the credentials, such as checking digital signatures and other proofs.

DID standards [50] defines DID subject as expressed using the id property in the DID document. DID documents record the mapping of subjects' identifiers and public keys of controllers. DID controllers are entities that are authorized to make changes to DID documents. In other words, the controllers can not only use the credentials of subjects, but also can decide who can control the identity. For some DID subjects, the DID documents might be managed by more than one DID controller. This configuration will often apply when the DID subject is an organization, corporation,

government agency, community, or other group that is not controlled by a single individual. This can happen in one of two ways, independent control and group control. In the independent control case, each controller has full power to update the DID document independently. In the case of group control, the DID controllers are expected to act together and reach an m-of-n consensus in some fashion, such as using cryptographic algorithms [68, 35] or an on-chain voting mechanism [20, 22].

2.2.3 Researches on Decentralized Identity System

Since DID is proposed, there are also some industrial productions, like [15, 31, 40], which allow users to gather credentials from issuers and selectively disclose them to verifiers. However, there are still some unresolved issues attracting attention from the researchers.

The privacy of credentials has attracted widespread attention from researchers. In 2022, Yamamoto et al. [71] propose Linked-Data-based verifiable credentials to enable selective disclosure among one or more verifiable credentials issued by multiple issuers. In 2022, De Salve et al. [12] describe an applicable method, allowing users to disclose only one attribute without invalidating the signature. CredChain [37] enables users to disclose attributes selectively based on the Merkle tree and redactable signatures. In 2020, De Salve et al. [18] propose a low-overhead selective disclosure method using hash functions. Zero-knowledge proof is also applied to achieve more precise information disclosure. SelfKey [15] is a platform that uses zero-knowledge proof (ZKP) protocols to minimize information leakage. D2CDIM [69] preserves identity privacy by the anonymous credential and zero knowledge in the cross-domain authentication scheme. In 2023, Rosenberg et al. propose zk-creds [53] protocol for flexible and privacy-preserving credential presentation. In 2023, Du et al. propose UCBlocker [14]. UCBlocker allows users to prove that they are qualified for the requirements of the verifiers without leakage of extra attributes.

In DID systems, the private key is important for a user to control their identities. Without private keys, the controllers cannot manage the identity, as well as credentials. In Web3, the problem becomes more serious. Loss of the private key translates to a complete relinquishment of identity control, encompassing digital assets and sensitive data. There are also some solutions to prevent the key lost. Firstly, in some systems, the users rely on centralized authorities to maintain their private keys. However, it necessitates users to place trust in these authorities implicitly. Secondly, some schemes [52, 73] enable users to create a mnemonic phrase based on their private key. The mnemonic phrase is stored in a secure medium. However, it remains susceptible to potential loss of the storage medium. Thirdly, in some systems[33], some committees are set to keep the users' private keys using secret sharing schemes. Users can regain their private keys by collecting fragments of private keys from committees. Even though a single committee cannot infer the private key information, concluded committees can control all the users' identities by gathering the fragments. Besides, social recovery [31] is a common solution, allowing users to authorize a set of guardians who can refresh or recover the users' secret keys. The users can specify delegates, instead of being forced to trust committees. However, if we directly apply the social recovery scheme in the multi-controller scenarios, it can lead to unnecessary overhead.

The above work related to DID is based on single-controller identities. Even though identity management for multi-controller scenarios has promising applications [50], little work has been done on identity management for multiple controllers. The uPort[31] allows multiple controllers to jointly control an identity, but only grants the same privileges to all controllers. Especially, in the uPort system, any controller can not only have permission for controller management, such as adding/removing a controller for the identities, but also have access to subjects' all credentials. Such a coarse-grained identity management scheme may lead to problems of identity impersonation and high key recovery overhead. impede the application of DID in multi-controller

scenarios. Besides, uPort also ignores the security and privacy issues arising from the unique characteristics of blockchain technology, i.e., data transparency and high ledger-commit latency. The missing consideration can lead to controller-correlation attacks and insecure controller revocation problems. Thus, it is necessary to design a fine-grained identity management scheme for multi-controller scenarios.

Chapter 3

MoDID: a Fine-Grained and Secure Identity Management Scheme for Multiple Controllers

3.1 Overview

In this chapter, we introduce MoDID, which is a fine-grained identity management scheme for issues caused by coarse-grained access control, i.e., identity impersonation attacks and high key recovery overhead. This chapter is structured as follows. The system model is described in section 3.2, including our hierarchical Identity scheme and flexible key recovery scheme. The implementation of our prototype platform and the experiment results are described in section 3.3. Finally, we conclude this paper in section 3.4.

3.1.1 Problem Statement

DID plays an essential role in the Web3 applications. Multi-controller scenarios have been added to the W3C DID standards [50] according to the common needs of joint control on a subject by multiple controllers. To cater to multi-controller scenarios, the uPort system [31] allows multiple controllers to jointly control an identity. However, it is coarse-grained, because all controllers are granted the same privileges. Especially, in the uPort system, any controller can not only have permission for controller management, such as adding/removing a controller for the identities, but also have access to subjects' all credentials.

Several problems in the existing systems impede the application of DID in multi-controller scenarios.

- **Identity impersonation**.An adversary controller can impersonate the identity without raising suspicion. This is possible because they can selectively remove other unsuspecting controllers, thereby gaining full control over the identity.
- **High key recovery cost**. In current key recovery schemes, all controllers rely on delegates to regain control over the identity, which brings extra gas consumption.

The root cause of these problems is coarse-grained identity management, which gives all controllers the same super permission. Therefore, it is necessary to design a fine-grained and low-overhead identity management scheme for multiple controllers.

3.1.2 Sketch of Solution

To address the above issues encountered by the existing system, in the paper, we propose MoDID, which complies with the DID standard proposed by W3C. MoDID is a fine-grained and low-overhead identity management scheme for multiple controllers by

hierarchical controller management and modifying the key recovery strategy. Firstly, for hierarchical controller management, we categorize controllers into three distinct roles based on their reliability, i.e., root controllers, delegates, and basic controllers [55]. Different controllers are responsible for different operations within the identity management process, such as controller management, recovering keys, and using credentials. Since root controllers possess the privilege to control the identity, we design two ways to manage them based on the collaboration of root controllers, i.e., the independent way and the group way. Secondly, to mitigate the key recovery cost, we devise a secure and flexible key recovery strategy. Specifically, root controllers can assist controllers in regaining control over their identity by substituting the addresses stored in the blockchain. Controllers resort to the social key recovery strategy, which incurs relatively higher overhead, only in instances where root controllers cannot execute controller management operations.

The major contributions of the work are summarized as follows.

- We propose MoDID, a novel fine-grained identity management scheme for multiple controllers. MoDID complies with the W3C DID standard. Our low-overhead solution allows multiple controllers to maintain reliable control over DID subjects by precise permission limits.

- We design a secure and flexible key recovery scheme for multi-controller scenarios. In MoDID, controllers can regain control through an optimized key recovery scheme to minimize identity loss while reducing the overhead.

- We build a DID prototype platform based on the Sepolia Ethereum Test Network. The result demonstrates the effectiveness of our proposed scheme in terms of gas consumption and transmission latency compared with existing systems, e.g., uPort.

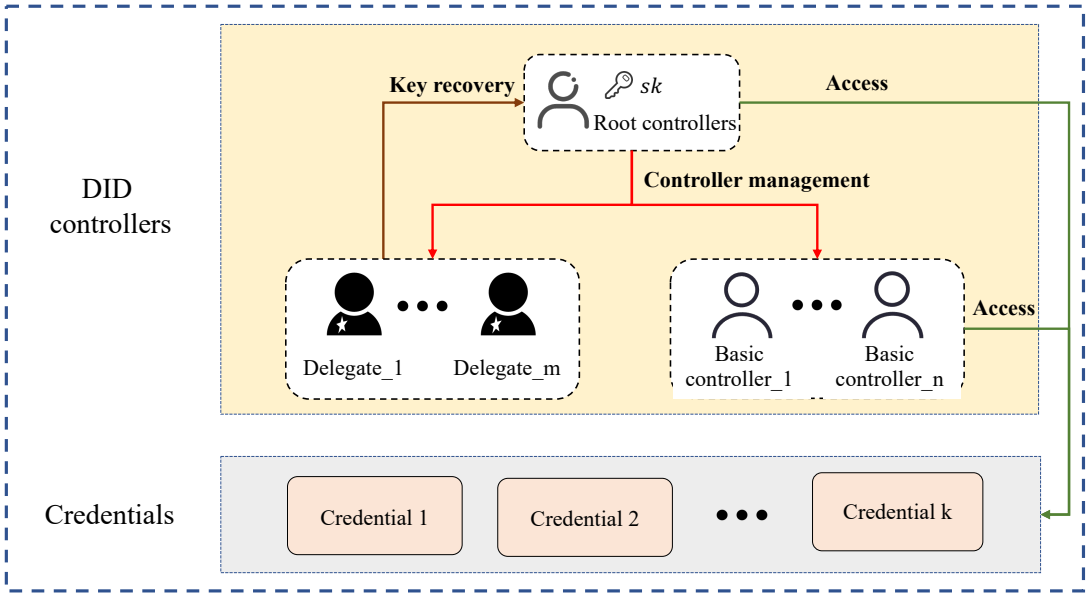


Figure 3.1: Hierarchical Identity Management Scheme in MoDID.

3.2 System Architecture

In this section, we introduce MoDID and our prototype platform for multiple controllers by hierarchical controller management. Since the process of identity management involves managing controllers and managing credentials, we divide our system into two parts, i.e., controller management and credential management, as shown in Fig. 3.2 and Fig. 3.4 respectively. In the controller management part, we use smart contracts in Ethereum to hierarchically manage controllers, like adding/deleting controllers or giving/removing controllers' roles. In the credential management part, we use smart contracts to store credentials' addresses and use the interplanetary file system (IPFS) to store credentials for scalability consideration.

Ethereum [6], introduced by Vitalik Buterin in 2015, is a decentralized blockchain platform that extends the capabilities of traditional cryptocurrencies like Bitcoin. Ethereum offers a programmable blockchain, enabling the development and execution of smart contracts [67] and decentralized applications (DApps) [7]. Another

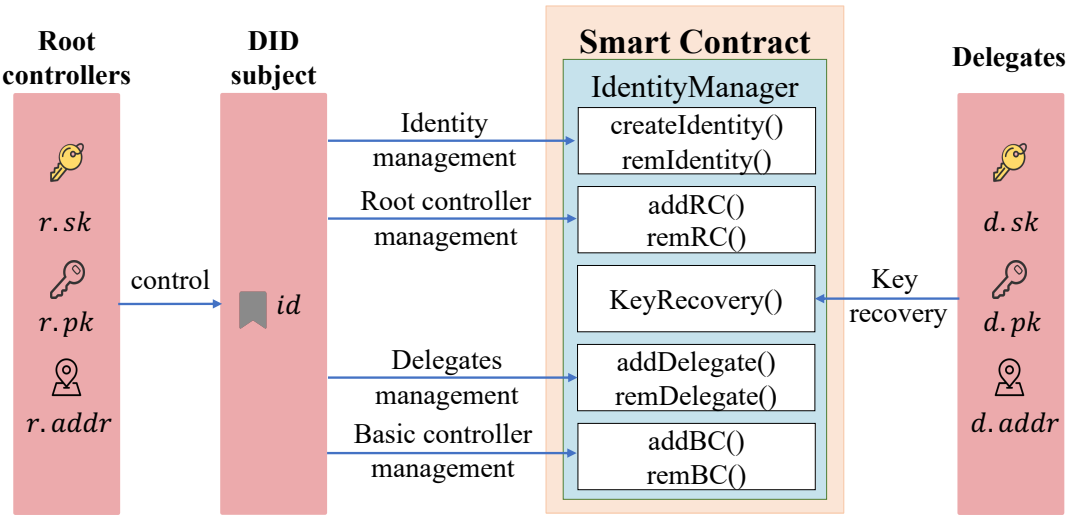


Figure 3.2: Identity Management Scheme in MoDID.

component in our system is IPFS. The IPFS is a distributed file system where peers connect to each other to store files[4]. It relies on hash-addressing to avoid redundant storage. The users are required to remember the hash addresses of their data for retrieval.

3.2.1 Hierachical Controller Management

We design a hierarchical controller management scheme to limit controllers' permission reliably, as shown in Fig. 3.2. According to the process of identity management, we classify controllers into three categories, i.e., root controllers (RC), basic controllers (BC), and delegates.

- RCs, which are usually executives in organizations, have the highest permission for identity. They can manage other controllers and all credentials. Especially, the controller management functions include adding/removing other controllers and giving controllers different permission based on their reputation. The credential management operations contain collecting, usage, and storage.

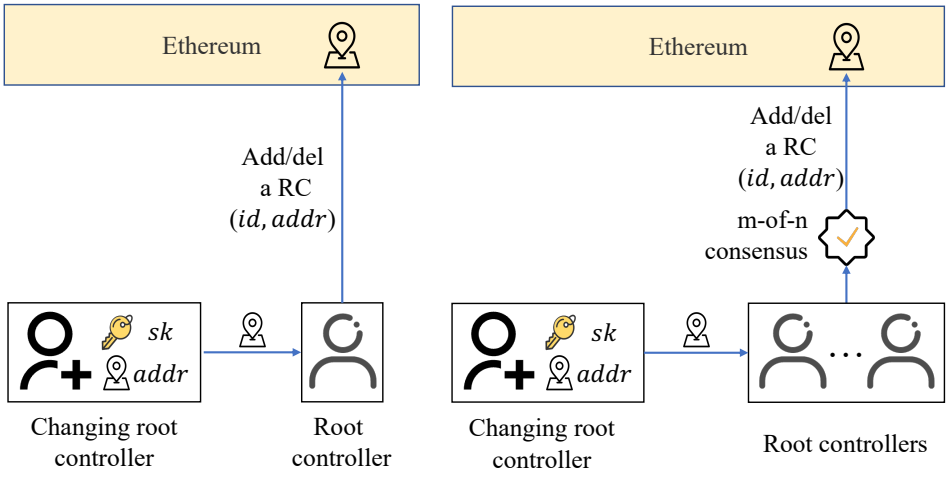
Table 3.1: Permission of Different Controllers.

	RC	Delegate	BC
Controller management	✓	-	-
Key recovery	-	✓	-
Update credentials	✓	-	✓
Present credentials	✓	-	✓

- Delegates are special controllers. Delegates can assist RCs in recovering control of their identity if RCs lose their private keys. The key recovery process will be described in 3.2.3.
- BCs, which are less trusted, are usually staff in an organization. BCs can not perform controller management operations, but they can present the credentials to verifiers.

From Table 3.1, we can see that the RCs have the privilege to control the identities. Thus, we design two ways for RCs management, i.e., independent way and group way, as shown in Fig. 3.3 (a) and Fig. 3.3 (b) respectively. This is because any malicious RC can optionally give any entity RC's permission or remove other controllers, which may lead to serious credential leakage problems. In the independent management scheme, any RC could independently do administration operations. While in a group management scheme, RCs must reach a m-of-n consensus using on-chain ways. Especially, only if more than θ_m RCs call the smart contract, the management operations could be performed successfully. θ_m is a threshold. Although group management may lead to extra gas consumption, it can significantly reduce the risk of identity impersonation and privacy leakage. This is because one RC can not optionally manage other RCs before reaching a consensus with other RCs.

MoDID improves the threshold for launching identity impersonation attacks by assigning distinct permissions to controllers based on their reliability. We assume that



(a) Independent Management Way for RC.

(b) Group Management Way for RC.

Figure 3.3: Two Ways for Root Controller Management.

there are r RCs, b BCs, and d delegates in a DID system. The probabilities that a root controller, basic controller, or delegate becomes malicious are denoted as P_r, P_b, P_d, respectively, where P_r is significantly lower than P_b and P_d. In traditional systems, since all the controllers possess super permission, the probability of an identity impersonation attack is $max\{P_b, P_r\}$. However, in MoDID, only RCs with high credit have the right to manage other controllers. Besides, to limit RCs' permission, we support both independent way and group ways for RCs. Thus, in the independent way, the probability of an impersonation attack is P_r. In the group way, the probability that they launch identity impersonation attacks is $P_r^{\lceil r\theta_m \rceil}$, where $\lceil \cdot \rceil$ denotes the ceiling function, indicating rounding up. Both P_r and $P_r^{\lceil r\theta_m \rceil}$ are lower than $max\{P_b, P_r\}$, which shows that MoDID effectively enhances system security against identity impersonation attacks.

The overview of our hierarchical controller management in our platform is shown in Fig. 3.2. It comprises two smart contracts, i.e., IdentityManager and Proxy. The IdentityManagement smart contract is used for controller management. It mainly

contains eight tasks, including creating/removing identity, adding/removing RCs, giving/revoking BCs' permission, managing the delegate list, and key recovery. The Proxy contract takes responsibility for identification and interaction with extra contracts.

The identityManagement smart contract contains functions related to controller management. These functions are represented as follows:

`createIdentity (root_controller, delegates, management)` is invoked by the RCs to create an identity. It takes the RC address, delegates list, and management way as inputs. Firstly, the controller generates a pair of keys based on Elliptic Curve Cryptography (ECC). To get the address, we can use the Keccak256 algorithm and get the hash value of the public key. The last 20 bytes of the hash value serve as the address. Due to the one-to-one binding relationship between the user address and the public key, in our system, the controllers are required to publicize their addresses. The delegates list contains the delegates' addresses. The management parameter refers to the management method of the root controller, including group way and independent way. After executing, it creates a Proxy instance. The instance address serves as the identification. Through the instance, controllers can call other contracts. After that, the root controller gains the privilege to control the identity. The root controller can invite other controllers to jointly control the identity by calling other related functions.

`addController (id, newController)` and `remController (id, oldController)` are executed by RCs to add/remove other root controllers. They both take identity and controller address to be processed as inputs. Especially, in a group management way, unless more than θ_m RCs call these functions and reach a consensus, operations cannot take effect. In an independent management way, any root controller can add or remove other controllers by calling the corresponding functions.

`addRecovery (id, newRec)` can be executed by RCs to add delegates. This func-

tion takes the identifier and delegate's address as inputs. After execution, the corresponding delegate can help the root controllers recover control over the identity when necessary. The opposite is `remRecovery (id, newRec)`, which is used to remove a delegate. Like `addRecovery`, it is executed by RCs. After execution, the revoked delegate cannot participate in recovering control over the identity.

`addBC (id, controllerAddr)` and `deleteBC (id, controllerAddr)`, which can only be revoked by RCs, are used for adding/removing BCs. They take the identifier of the subject and the address of BC as input. After `addBC()` function execution, the corresponding BC can present the credentials of the subject. After revoking function `deleteBC()` execution, the revoked BC cannot control the identity by presenting its credentials.

3.2.2 Verifiable Credential Management

In this part, we introduce our verifiable credential management scheme, including collection, usage, storage, and verification. The structure of credential management is shown in Fig. 3.4. We use the credentialManagement smart contract for 1) storing credential storage addresses, 2) checking controllers' permission, 3) verifying the signatures of credentials. As for credential storage, we take advantage of the IPFS system due to the expensive gas fees of Ethereum.

Like physical credentials, verifiable credentials are issued by authorities to prove subjects' attributes. Verifiable credentials contain three parts, i.e., metadata, claims, and signatures. Metadata contains properties of the credential, such as the issuer's identity, the expiry date, and the UUID. The UUID is the unique identifier of credentials. Claims are expressed as key-value pairs, like `{degree: Bachelor's degree}`. Issuers' signatures ensure that the credentials are unmodified and issued by authorities.

In a verifiable credential system, there are three parts, i.e., issuers, holders, and verifiers[63]. Issuers are usually authorities, like governments or universities. Holders

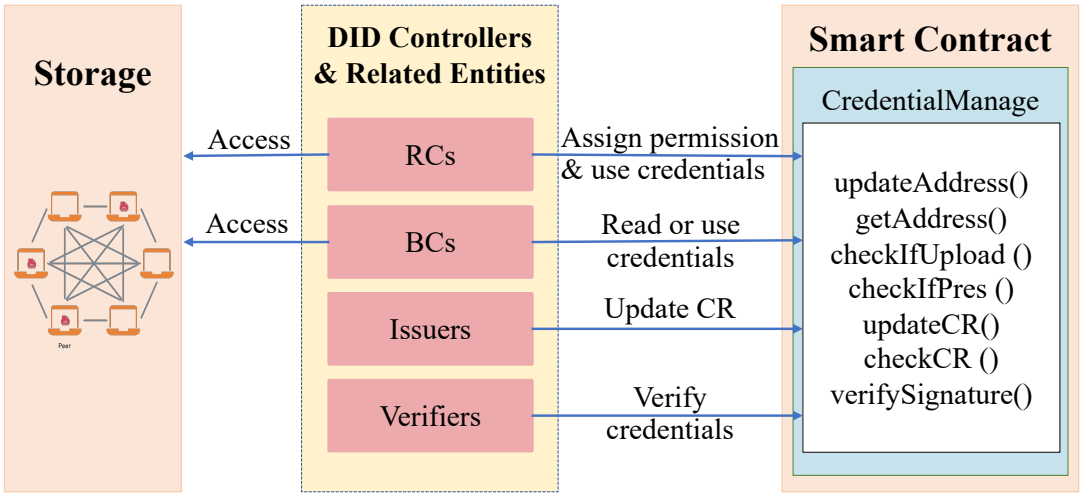


Figure 3.4: Credential Management Scheme in MoDID.

can present the verifiable credentials to verifiers. Verifiers will verify the validation of the Verifiable credentials for identification or authentication. To be specific, when holders want to prove their attributes to verifiers, holders request issuers to issue verifiable credentials containing the attributes proof. Then, holders must sign with their private keys to prove their ownership over the identity. Apart from integrity, the signature can make the credential non-transferable. This is because other entities without the correct private keys cannot forge the signature and cannot forge a correct signed verifiable credential. The signed verifiable credential is then presented to verifiers. The signed verifiable credential is called a verifiable presentation. Here is an example of a verifiable presentation to prove the subject's attribute of a bachelor's degree.

{

'uuid': 'ec8910ad-b2d3-49ad-b27d-b73ed8bc9328',

'type': 'credential_type',

'issuerId': '0x063103A47e4bd93A407d1435818d294408BAD44b',

‘issuanceDate’: ‘2023-07-04 14:15:00’,

‘expirationDate’: ‘2023-07-11 14:15:00’,

‘credentialSubject’:

{‘id’: ‘0xcB4e66eCA663FDB61818d52A152601cA6aFEf74F’,

‘claim’:

{‘degree’: ‘bachelor degree’,

‘university’: ‘Hong Kong Polytechnic University’

}

},

‘proof of issuer’:

{‘type’: ‘ECDSA’,

‘verificationAddress’: ‘0x53d72B5B1Ba635D5fc03180701D25119Be91CB73’,

‘signature’:HexBytes(‘0x9f62062782015c77391bc507ac6416d2beaff380859425d

4aef80149b5b425e0408b049a4fcc317153551836267d59b48ac34124d3857acec05998

1b37a4b02a1c’)

},

‘proof of holder’: {

‘type’: ‘ECDSA’,

‘verificationAddress’: key_1,

‘signature’:HexBytes(‘0xa7011ae8435dc21e37d64630a842f1b0694332103eebf0ee

1b019f3361706ee17ca3984377e3a41fd717d94114f3f3557a625d285cce75977e87d56fb

```
2b19fd91b')
  }
}
```

After receiving the verifiable presentation, verifiers will check the validation of the verifiable presentation, including 1) checking the expiry time, 2) checking revocation information, 3) verifying the controller's access, 4) verifying the signatures of holders and issuers.

Apart from the usage of credentials, MoDID also supports credential revocation operations. If the users misbehave or their credentials are at risk, issuers can revoke users' credentials. The issuers can upload the revoked UUIDs to the smart contract, due to the public nature of the smart contract. They can call the `setRevo` function to publicize the UUID of revoked credentials. Anyone can check whether a credential is revoked or not by querying it.

Credential storage is vital for controllers. This is because using credentials is a frequent operation, while pre-operations like issuing and signing for the credentials are time-consuming. Thus, we support the holders in uploading the encrypted credential presentation to the IPFS. The IPFS is a distributed file system where peers connect to each other to store files [4]. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting IPFS hosts. The file system is maintained by many nodes and thus reduces The risk of single-point failures. The decentralized characteristic improves the scalability of the MoDID system. Especially, RCs can use a shared secret key to encrypt the signed credentials. They can use any Data Encryption Algorithm, like Data Encryption Standard (DES) or Advanced Encryption Standard (AES). After the access check by the smart contract, the RCs upload the credentials to the IPFS. Since the IPFS rely on hash addressing, the users can upload the hash address of their credentials to the smart contract to reduce the loss

of credentials. Note that, since multiple controllers have access to credentials, the signatures in the credentials may be from different controllers.

The credentialManagement smart contract presents the following functions:

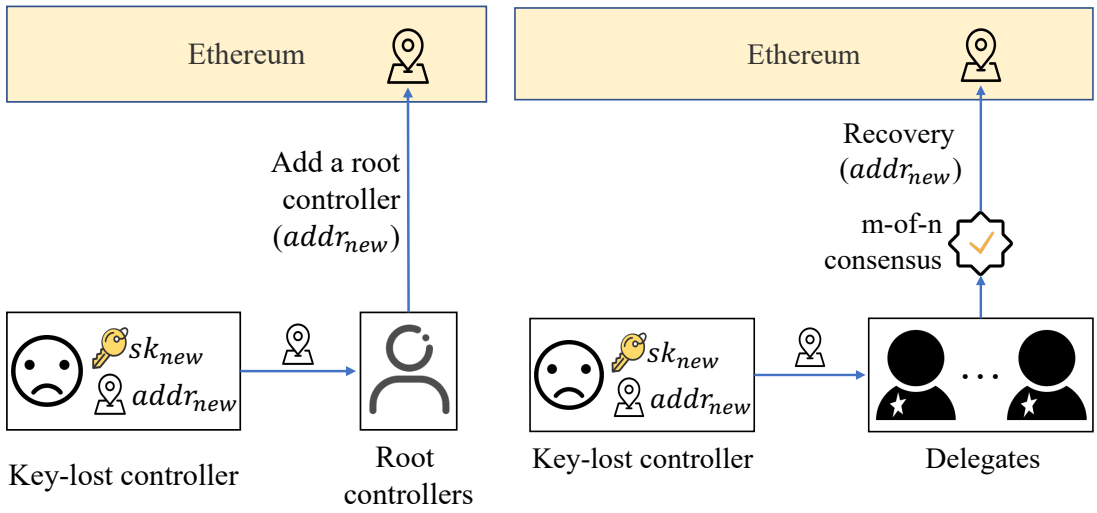
`logUpdate (id, type,fileAddr)` function will be invoked by RCs to store users' credentials' addresses. The first input is a string to allow users to store different credentials partly. The second input is the hash address of credentials in IPFS. After the successful invocation, the mapping between the subject's identifier, the type of credential, and the credentials' storage address is recorded in the blockchain system. The controllers can get the storage address if necessary. `getCreAddr (id, controller, type)` is invoked by the corresponding controllers with permission to get the corresponding hash address. After getting the hash address, users can download the encrypted credentials from the IPFS.

`setRevo (UUID, cred)` function can be invoked by issuers to revoke credentials. The smart contract will check the validation of the signatures and then record the UUID. The opposite is `checkRevo (UUID)`, which anyone can query to check whether the credential is revoked or not.

3.2.3 Key Recovery Scheme

In a DID system, users are responsible for properly storing private keys. The private key loss means the complete loss of the identity, including private credentials and digital assets. Thus, we design a flexible and low-overhead way for key recovery, considering credential revocation and network delay. Like authentication systems in Web2, we consider control recovery to help controllers recover control over their identity and credential revocation (CR) after recovering controllers' control to reduce the risk of identity impersonation.

In the control recovery part, we discuss two situations based on whether the absence of



(a) RC-assisted Control Recovery. (b) Social Control Recovery.

Figure 3.5: Two Recovery Mechanisms for Key-Lost Controllers.

the key-lost controller will affect the execution of controller management operations. 1) If the key-lost controllers are BCs, delegates, or RCs without whom the rest of the RCs can still reach managing m-of-n consensus, the RCs can directly replace their address to control the identity. We call this method RC-assisted control recovery, as shown in Fig. 3.5 (a). Especially, the key-lost controller locally generates a new pair of keys and informs RCs of their new addresses in an off-chain way. RCs can replace the original address with the new one by calling `remController` and `addController` functions so that the key-lost controller can use the newly generated key to control the identity.

2) We also make use of social control recovery in some cases, as shown in Fig. 3.5 (b). Since it is high-overhead, we only rely on it for control recovery in the two cases. Firstly, in group management, the rest of the RCs can not reach a m-of-n consensus because of the absence of the key-lost RC. Secondly, in independent management, without the key-lost RC, no other controllers have the right to add a new RC. In social control recovery, we rely on delegates. Delegates, usually controllers' friends

or relationships, can help RCs recover control over their identity. Especially, when RCs lose their private keys, RCs can generate new key pairwises. Then, RCs inform delegates of the new address in off-chain ways. After receiving the new address, delegates can call `Recovery` function with the input identifier and new address. If more than θ_r delegates call the function, the key-lost RCs can use the new private key to control the identity again. θ_r is a threshold.

However, in an asynchronous environment, it is common for delegates to repeatedly call the function to add different addresses as the identity root controller. Therefore, we set a time threshold T. Within time T, only one controller can be added through the key recovery scheme. If delegates successfully recover the private key for the identity id in time T, the system will refuse any control recovery request from any delegates.

After recovering controllers' control, CR operations can reduce the risk of identity impersonation. This is because anyone who stole the key may access credentials stored in the IPFS, leading to privacy leakage and identity impersonation problems. In MoDID, we consider controllers except for delegates, because delegates without access to credentials cannot lead to credential leakage. Besides, rather than revoke all credentials, we only revoke credentials signed by controllers except the key-lost controller. This is secure because credentials signed by the key-lost controller are invalid without the other controllers' private keys. Especially, controllers first send revoking requests to corresponding issuers. Secondly, issuers publicly store the UUID in the smart contract after verification.

3.3 Platform Implementation and Evaluation

In this section, we build a DID prototype platform based on the MoDID algorithm. Besides, in order to make our protocol platform user-friendly, we also build a user

interface (UI), which serves as a digital wallet. In our prototype system, the issuers can create the issuers' identity and issue credentials to the users. The users can create identities, manage the controllers with fine granularity, and present credentials. The verifiers can also verify the credentials presented by holders. To validate the feasibility and practicality of MoDID, we compare the performance of MoDID and the uPort system in various tasks. Firstly, we deploy our smart contract in the Ethereum Sepolia Testnet. Secondly, for hierarchical controller management operations, rather than time consumption, we evaluate the gas consumption, which measures the computation of operations. This is because time consumption can be significantly reduced if the users pay more gas fees to miners, while gas consumption is a relatively stable measurement. Thirdly, We use time consumption to measure credentials management operations, i.e., storage and verification. To prove the effectiveness of our scheme, we implement the popular uPort system as the baseline approach.

3.3.1 MoDID Implementation

We implement MoDID using the Ethereum blockchain and the IPFS system. The experiments are conducted on a MacBook Air with Apple M2 CPU and 8 GB of RAM. We use Node 18.13.0. For the Ethereum system, firstly, we use Solidity 0.8.17 to compile the three smart contracts. Subsequently, we deploy and test the smart contract on the Sepolia Ethereum Testnet using Truffle 5.7.2 and Infura. Truffle achieves built-in smart contract compilation, linking, and deployment. Additionally, the codes are written in Jupyter Notebook using Python 3.10.9 for other functions. We use the Python library Web3 5.31.3 to interact with Ethereum and use the Python library Ipfshttpclient 0.4.13.2 to update/download credentials in the IPFS system. Besides, in order to make our protocol platform user-friendly, we also build a digital wallet using Flask 1.1.2 for the backend interface. We use Web3.js and Vue.js for the front page design.

Table 3.2: The Gas Used for Smart Contract Deployment.

	Gas used	Total cost (wei)
IdentityManage	5,341,907	96,154,326
Proxy	1,342,453	24,164,154
CredentialManage	1,956,325	3,521,385

In our DID ecosystem, there are 1,000 DID subjects. We test our system using arbitrary subjects. For the identity, there are 3 RCs, 35 BCs, and 3 delegates. There are 200 credentials portraying the attributes of the DID subject. RCs can manage other controllers, upload credentials to the IPFS, and present all credentials to verifiers. Since RCs' privilege, we support two ways for RCs management, i.e., group and independent management. In group management, RCs must reach a consensus before admin functions take effect. While in independent management, every RC could manage other RCs independently. Delegates take responsibility for key recovery through on-chain voting way. BCs can only read and use credentials in the IPFS. In our setting, the admin threshold θ_m and key recovery threshold θ_r are 2/3. Only if more than 2 RCs/delegates vote do the controller management operations/key recovery operations take effect. For credential revocation, issuers will update revoked UUIDs to smart contracts every 30 minute.

3.3.2 Performance Analysis

We deploy the smart contract on the Sepolia Testnet. Table 3.2 displays the creation transaction details of the three smart contracts, identityManagement[1], proxy[2], and credentialManagement[3] respectively in the Sepolia Ethereum test network.

[1]https://sepolia.etherscan.io/address/0xCfEB869F69431e42cdB54A4F4f105C19C080A601
[2]https://sepolia.etherscan.io/address/0x5b1869D9A4C187F2EAa108f3062412ecf0526b24
[3]https://sepolia.etherscan.io/address/0xe78A0F7E598Cc8b0Bb87894B0F60dD2a88d6a8Ab

We compare the gas consumption with that of the uPort system. Table 3.3 shows the comparison. Firstly, from the comparison of the independent management operations, we can see that the gas consumption of some functions is lower than that of the uPort system, including adding/removing an RC, adding/removing a delegate, storing credential addresses, and key recovery. Creating identity operations, less frequently used than other functions, costs more gas. This is because more traversal statements are used to reduce the gas consumption of other functions. Secondly, the group management and BC management functions are especially used in MoDID. The group management scheme costs higher gas consumption than that of independent management. However, the group management way can significantly reduce the risk of identity impersonation. This is because RCs are required to reach an m-of-n consensus in on-chain ways. BC management functions are used to add or remove BCs, with gas consumption $5,695$ and $8,490$, respectively. Because MoDID consumes less gas than the same operations in the uPort system, we can infer that our unique functions for multi-controller scenarios are low-consumption.

As for the verifiable credential system, we evaluate the time consumption for storage and verification. We first compare the transmission latency of uploading/downloading credentials for storage with that of the uPort system. Because the users upload/download text credentials files with small sizes, we use credential files ranging from $894Byte$ to $103KB$ for testing. The result is shown in Table 3.4. When the sizes of credentials are $894Byte$, $2KB$, and $103KB$, the uploading/downloading delay is lower than that of uPort. If the sizes are $7KB$ and $12KB$, the uploading delay is lower, while the downloading delay is slightly higher. The transmission latency of uploading/downloading in MoDID is similar to that of baseline on average, which implies that the latency caused by credential transmission is acceptable.

We also test the efficiency of credential verification. In traditional schemes, verifiers check the expiration, revocation, and signatures. While in MoDID, verifiers need one additional operation, i.e., controllers' permission check. We compare the time

Table 3.3: Gas Consumption Comparison.

	Functions	MoDID	UPort system
Independent management	Create an identity	1,418,356	748,422
	Add a RC	34,319	85,447
	Remove a RC	17,240	48,591
	Add a delegate	34,493	36,046
	Remove a delegate	17,016	17,160
	Key recovery	166,245	202,486
	Add a BC	5,695	-
	remove a BC	8,490	-
Group management	Create an identity	1,350,036	-
	Add a RC	128,733	-
	Remove a RC	112,404	-
Credential	Upload credential address	75,001	84,830
	Revoke credentials	52, 477	-

consumption of credential verification in MoDID with the traditional scheme. Fig. 3.6 plots the variation of time consumption with the number of revoked credentials. The average time consumption of credential verification increases from $0.0441s$ to $0.0528s$ because of the addition of permission checks. Although the operations bring an extra average $0.0086s$ time consumption, it reduces the risk of identity impersonation and privacy leakage.

MoDID has good scalability. First, the decentralized nature of identity management enables MoDID with promising scalability. We use Ethereum to store users' mapping of identifiers and addresses. The users rely on IPFS to store the credentials. The IPFS is a distributed file system with low uploading/downloading latency. Multiple nodes jointly maintain identity-related data. In large-scale scenarios, when some nodes fail,

Table 3.4: Transmission Delay of Credential Storage.

Credential	MoDID		UPort system	
size	Uploading	Downloading	Uploading	Downloading
2 KB	6.9260 ms	6.1218 ms	7.9848 ms	7.4203 ms
7 KB	8.9459 ms	8.2831 ms	8.9601 ms	8.1121 ms
12 KB	9.8459 ms	8.0857 ms	9.4342 ms	8.4042 ms
103 KB	121.5538 ms	17.2641277 ms	123.5351 ms	18.7418 ms

the whole system can also function normally. Secondly, mechanistic security makes MoDID able to be applied to large-scale scenarios. We propose a hierarchical identity management scheme, enabling fine-grained controller access control and optimized key recovery mechanisms. MoDID resists impersonation attacks and reduces the key recovery overhead. Thirdly, through the experimental results, we can conclude that MoDID is low-overhead. MoDID consumes much less gas consumption than state-of-the-art. Even though the time consumption of verification is slightly higher than that of traditional models, it makes our system resist impersonation attacks caused by malicious or misbehaving controllers. Thus, MoDID has promising scalability.

3.4 Chapter Summary

In this chapter, we propose MoDID, a fine-grained and W3C-standard compatible identity management scheme for multiple controllers. To address the identity impersonation issues not considered in existing schemes, MoDID establishes a hierarchical identity management system, allocating different levels of credential access and controller management permission to distinct controllers. Besides, MoDID proposes a flexible key recovery scheme for multi-controller scenarios with less overhead than existing systems. We also develop a prototype system to validate the feasibility and

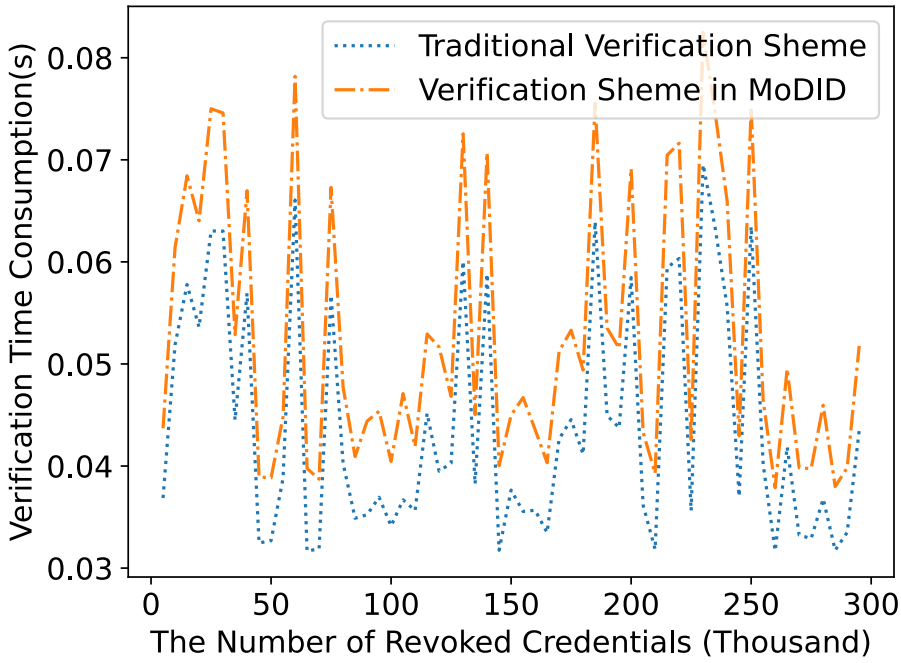


Figure 3.6: Time Consumption Comparison of Credential Verification.

practicality of MoDID. The evaluation shows that MoDID allows multiple controllers to securely control identities while incurring less time and gas consumption than the uPort system.

Chapter 4

Privacy and Secure Enhancement against Attacks in Multi-Controller Scenarios

4.1 Overview

In Chapter 3, to solve the problems caused by coarse-grained identity management, we propose MoDID, a fine-grained and low-overhead identity management scheme. However, the security and privacy issues have not been completely addressed. In this chapter, we focus on the issues stemming from blockchain's characteristics, i.e., data transparency and high ledger-commit latency. We find two attacks for multi-controller identities caused by the blockchain's characteristics, i.e., controller-correlation attacks and identity impersonation attacks by revoked controllers. We also propose a privacy-preserving and secure decentralized identity management scheme to resist the two attacks. This chapter is structured as follows. In Section 4.2, we give a detailed description of our proposed problems in multi-controller scenarios, i.e., controller-correlation attacks and identity impersonation attacks by revoked controllers. To

mitigate controller correlation attacks, we present a privacy-preserving scheme and its detailed designs in Section 4.3. To reduce the risk of identity impersonation attacks, we propose a lightweight identity authentication scheme, which is introduced in Section 4.4. Finally, we conclude this chapter in Section 4.6.

4.1.1 Problem Statement

DID is a robust and privacy-preserving identity management scheme in Web3 applications. In 2022, W3C defines the DID application to multi-controller scenarios. There is research on decentralized identity management in multi-controller scenarios. The uPort [39] allows multiple controllers to control an identity. But uPort is coarse-grained on privilege allocation. Based on the uPort system, we propose MoDID in Chapter 3, which is a hierarchical identity management scheme with fine-grained controller access control. controllers are divided into three distinct roles based on their reliability, i.e., root controllers (RC), delegates, and basic controllers (BC). Different controllers are responsible for different operations within the identity management process, such as controller management, key recovery, and credentials presentation. Despite these advancements, current solutions do not pay attention to the prevalent linkability of controllers and the demanding real-time retrieval of public keys in multi-controller scenarios. The root cause of this oversight lies in the failure to consider the characteristics of blockchain systems, namely data transparency and high ledger-commit latency.

The missing consideration of the controller likability and real-time retrieval requirement may lead to serious privacy and security problems.

- **Controller-correlation attacks**. It is common practice for controllers to use a single key pair to control multiple subjects in traditional solutions. The bad thing is that controllers directly upload the plaintext of public keys to the blockchain[50]. Due to the data transparency nature of blockchain systems, the

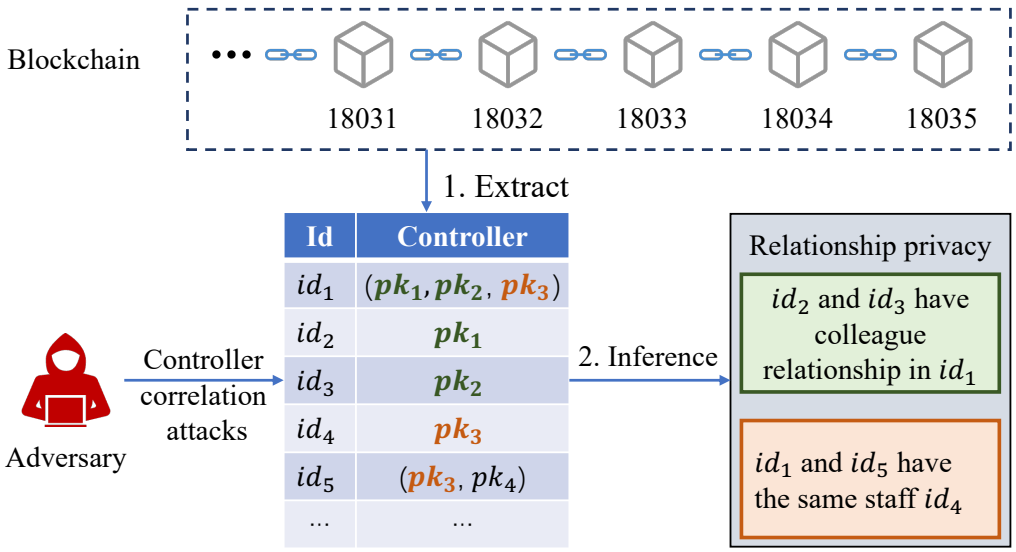


Figure 4.1: An Example of Controller-correlation Attacks.

public keys are linkable. This insufficient consideration leads to the leakage of user privacy. Adversaries can correlate the public keys and infer relationships between different subjects, like colleague relationships, as shown in Fig. 4.1.

- **Identity impersonation attacks by revoked controllers**. The verifiers are assumed to retrieve real-time public keys for verification, which is impracticable. The verifiers with capability-limited devices experience long-time block synchronization latency to synchronize blocks. They may retrieve stale public keys in the verification process [28, 24]. Consequently, the revoked controllers can impersonate the identity, and access illegal resources, as shown in Fig. 4.2. This problem becomes more severe for multiple-controller scenarios due to frequent revocation operations on the controllers.

4.1.2 Sketch of Solution

To address the above issues encountered by the existing system, we propose a privacy-preserving and secure identity management scheme for multiple controllers, which

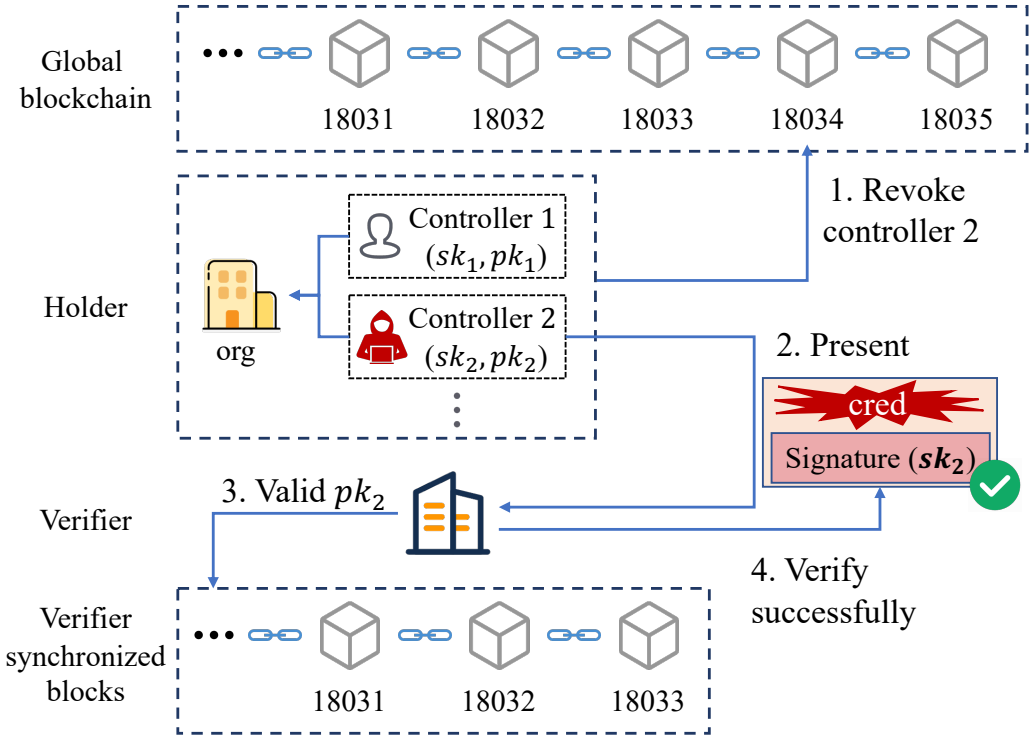


Figure 4.2: An Example of Identity Impersonation Attack by Revoked Controllers.

complies with the W3C DID standards [50]. We can prevent external adversaries from inferring the relation between different DID subjects. Besides, our system enables the verifiers to access fresh public keys without compromising system availability. To resist to controller-correlation attacks, we design a privacy-preserving controller management scheme based on the Merkle tree. Specifically, the controllers of multi-controller identities locally construct a Merkle tree containing the controllers' public keys and permission. Subsequently, they upload the digest of the Merkle tree to the blockchain. By concealing controllers' public keys behind the digest, adversaries are unable to deduce relationships between identities by correlating controllers' public keys. To solve the insecure controller revocation problem, in our scheme, the verifiers only download lightweight block headers regularly, rather than synchronizing heavy blocks. The holders take responsibility for providing the freshness proof, proving that their digest is included in the newest block. With the proof and block

headers, the verifiers can access up-to-date public keys efficiently, reducing the risk of impersonation attacks from revoked controllers.

The major contributions of the work are summarized as follows.

- We find two new attacks caused by multiple controllers toward DID management, i.e., controller correlation attacks and impersonation attacks caused by controller revocation. These attacks may lead to serious privacy leakage and insecure authentication problems.
- We propose a privacy-preserving and secure identity management scheme for multiple controllers to resist the two attacks. Our scheme allows the controllers to prove ownership over the identities without publicizing their public key. Besides, our secure authentication strategy allows the verifiers to access up-to-date public keys without synchronizing heavy blocks
- We build a DID prototype platform based on the Sepolia TestNet. The result demonstrates the effectiveness of our proposed scheme in terms of gas consumption, and cost of proof generation and verification compared with the state-of-art work.

4.2 System Design

4.2.1 System Model

Our system contains four different roles. We summarize the roles and their functions in the following.

- **Ethereum system**: The Ethereum system supports smart contracts. It serves as a public ledger to maintain the mapping between users' identifiers and public

keys. The parties participating in the authentication process, as well as any external parties, can fetch the public key with the input identifiers.

- **Issuer**: Issuers refer to the centralized authorities, like governments or universities. These authorities can issue verifiable credentials to the holders.
- **Holder**: The holders are usually users in the Web3 era. They can collect verifiable credentials from issuers and keep these credentials in their own wallets. If they want to get authentication, they can present the verifiable credentials to the verifiers. The presented verifiable credentials are also called verifiable presentation.
- **Verifier**: The verifiers are usually web service. They participate in the authentication process by verifying the verifiable presentation presented by the holders. Apart from checking basic information, the verifiers are required to fetch public keys from the blockchain system, and then verify the signatures of both issuers and holders. After credential verification, the verifiers can provide web resources for the qualified holders.

4.2.2 Threat Model

We find two attacks that may arise in the existing DID systems, which impede the application of DID in multi-controller scenarios. In this section, we describe the two attacks, i.e., controller correlation attacks, and identity impersonation attacks.

Controller correlation attacks

We assume that the adversary can access the data recorded in the blockchain. The adversary's goal is to infer the relationship privacy between different subjects by correlating the controllers' public keys in the blockchain. We assume that the verifiers are not curious about the relationship privacy. We also assume that the adversary

can not eavesdrop on the communication between the holders and verifiers due to the high cost.

The adversary can implement controller-correlation attacks. This is because the controllers upload the plaintext of $mapping(id, pk)$ to the blockchain. The controllers may use a single key pair to control multiple subjects. The adversary can fetch public keys recorded in the blockchain and infer the relationship between different identities by linking public keys. Fig. 4.1 illustrates an implementation example of the proposed controller-correlation attacks. By correlating pk_1 and pk_2, the adversary may infer that the id_2 subject and the id_3 subject are colleagues in the organization with id_1. Besides, by correlating pk_3, the adversary can infer that organizations with id_1 and id_5 have the same staff with id_4.

Identity Impersonation Attacks

We assume that the adversary previously had control over the multi-controller identity, but this access was recently revoked. The adversary's goal is to impersonate the identity and enjoy the illegal resources, even though he is removed. Besides, we assume that the verifiers are full nodes and take part in the ledger-commitment process.

The adversary can launch the identity impersonation attacks. In DID systems, controllers' public keys are recorded in the blocks. Due to high block synchronization latency, the verifiers with capability-limited devices may experience long-time latency in synchronizing the latest blocks. Thus, the verifiers may have difficulty in retrieving the real-time public keys. Due to the stale public key retrievals, the adversary may impersonate the identity after revocation and enjoy illegal resources. Fig. 4.2 shows an example of identity impersonation attacks. The revocation operation of controller 2 is executed in block 18034. After the emergence of block 18035, controller 2 presents the organization's credential to the verifier with only the first 18033 blocks. Despite

the revocation, the verifier incorrectly validates the credentials. This is because, without the latest two blocks, the verifier mistakenly considers that controller 2 still retains control over the identity.

4.2.3 Use Case

Our system provides a strong privacy and security guarantee. Thus, our system has a wide range of use cases. We list several examples of what our system brings to the Web3 ecosystem.

Know Your Customer (KYC). Laws in various jurisdictions require service providers to meet strict compliance responsibilities to identify or qualify their customers, for example, through KYC [38]. Authentication often requires the user to provide personally identifiable information (PII). However, there are no universal identifiers for authentication. Credentials with different schemas can increase the burden of the verifiers. Our system serves as a decentralized and universal identity management scheme, which makes it possible for users to use universal identifiers and credentials for authentication.

Decentralized social networks in Web3. Apart from the web services, the users may also need authentication and control of access to build healthy social networks[1]. Customizable access control is important in the Web3 era. For example, it is a common need for users to block unwanted calls from unknown numbers [14] or block spam from unknown senders [1]. Traditionally, centralized service providers, like telephone operators or email operators, can classify wanted or unwanted social requests using artificial intelligence technologies [29]. However, the centralized strategy is difficult to apply directly to the Web3 era, due to its potential privacy leakage and data centralization. Secondly, achieving customizable access control leads to high overhead. In our system, everyone can define individual access policies using low overhead. They

[1] https://www.kaleido.io/blockchain-blog/use-cases-for-decentralized-identity.

can verify the provided credentials using the public keys recorded in the blockchain system and authenticate the other individuals' identities easily in a decentralized and privacy-preserving manner.

4.3 Privacy-preserving Scheme for Controller Correlation Attacks

Existing systems for multi-controller scenarios may lead to controller-correlation attacks. In our system, we propose a privacy-preserving scheme based on the Merkle tree for the controller registry. Our scheme allows controllers to prove their ownership over an identity without publicizing their public keys.

The Merkle tree algorithm [34] is proposed by Ralph Merkle in 1979, which is a vector commitment scheme [45]. The algorithm is widely applied in blockchain systems for its efficient data verification and privacy preservation. The basic structure of the Merkle tree is the tree data structure. It can be a binary tree or a polytree. Here, we use the binary tree as an example. We define the input of the Merkle tree as vector $V = (m_1, m_2, ..., m_n)$, which can be data with arbitrary length. Firstly, to get the leaf nodes, we compute the hash value of each element m_i. Due to the one-way characteristic of the hash function, it is difficult to infer m_i based on the hash value. Secondly, the Merkle tree hashes all elements in V and then recursively hashes the result in a pairwise manner using the defined hash function until obtaining a single element called Merkle root or digest. The Merkle root serves as the commitment of the vector. Given the Merkle root, We can determine whether an arbitrary element is contained by vector V in an efficient and privacy-preserving manner. For membership proof, we only obtain the sibling nodes of the targeting leaf nodes. Opposite, we compute the hash value of the Merkle proof in a pairwise manner and compare the computed digest with the given one. Thus, The time complexity of proof generation and veri-

fication is both $O(logn)$. The Merkle tree can also be used for privacy preservation. Besides, it can also be used to protect users' privacy. This is because it allows users to certify that potential candidates are an element of a certain set without revealing the other members of the set.

We consider a hybrid DID system, including single-controller identities and multi-controller identities. For single-controller identities, the controller just uploads the plaintext public keys to the blockchain, because they are not linkable. For multi-controller identities, the controllers can build a pk Merkle tree containing the controllers' public keys. They only upload the digest of public keys to the blockchain. Due to the one-wayness of the hash function, adversaries cannot infer public key information from the digest.

Specifically, for multi-controller identities, the controllers locally build pk Merkle tree and upload the digest to the blockchain system.

- *Local digest computing.* We define the controller vector $V = \{v_1, v_2, ..., v_n\}$. To be compatible with the hierarchical scheme in the MoDID system, we define controller i using $v_i = \{pk_i, r_i\}$. pk_i is the public key. r_i is the role of the controller. Controllers with different roles have distinct access to identity management operations. With V as input, the controllers build the pk Merkle tree. Firstly, the controllers compute every leaf node. $leaf_i = hash(v_i)$. They can use arbitrary hash functions. Here, we use the SHA256 algorithm as an example. The controllers hash all leaves and recursively hash the result in a pairwise manner until obtaining a single element called digest or Merkle root. With the pk Merkle tree, any controller can efficiently prove their ownership over the identity without leaking other controllers' public keys.

- *Digest upload.* After obtaining the digest, any controller i owning high permission with v_i uploads the mapping to the smart contract. Firstly, the controller i calls the corresponding function in the smart contract. The controller is also

called smart contract caller. Apart from identifier and digest, the smart contract caller is required to provide Merkle proof ρ_{pk} of the pk Merkle tree. The Merkle proof proves that v_i is a member of V, which means that the smart contract caller possesses the privilege to control the identity. Specifically, the Merkle proof is the hash value of the encountered sibling nodes along the path upwards, starting from the target leaf node. Then, the smart contract verifies Merkle proof ρ_{pk} with the given digest, and then records the $mapping(id, digest)$ in the blockchain.

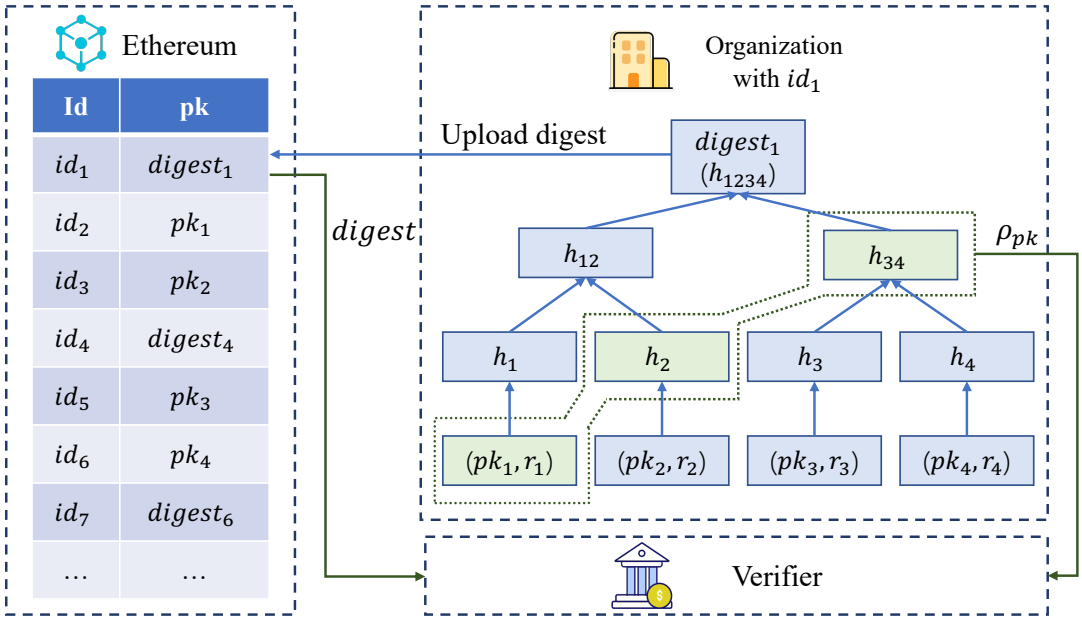


Figure 4.3: A Privacy-preserving Scheme for Controller-correlation Resistance Based on Merkle Tree.

Fig. 4.3 shows an example of the registry process of an identifier with four controllers. The controllers construct a pk Merkle tree with input $\{(pk_1, r_1), ..., (pk_4, r_4))\}$. Then, they upload $digest_1$ to the blockchain. If the controller with (pk_1, r_1) wants to prove their ownership over the identity, they can provide the Merkle proof $\rho = \{h_2, h_{34}\}$ for ownership verification. After that, the controllers can use the identity for authentication by providing the membership proof ρ_{pk} of the pk Merkle tree.

We also support controller changes. 1) Controller addition. If the controllers want to invite other controllers with (pk_j, r_j) to jointly control the identity, they can modify the local Merkle tree. Then upload the newly generated digest to the blockchain. After execution, the controller with (pk_j, r_j) can control the identity with corresponding permission. 2) Controller revocation. It is similar to controller addition operations. The controllers remove the corresponding leaf nodes in pk Merkle tree and obtain a new digest. Then the controllers upload the newly generated digest to the blockchain.

Security analysis. Our system can resist to controller-correlation attacks. In our system, the controllers locally build the pk Merkle tree by recursively hashing nodes. They only upload the digest to the blockchain. The one-wayness of the hash function means that we cannot infer public key information from the digest. Thus, the adversaries can not infer a relation between different identities by correlating the public keys recorded in the blockchain.

4.4 Identity Authentication Scheme with Secure Controller Revocation

In Section 4.3, we propose a privacy-preserving scheme based on the Merkle tree structure for controller-correlation attacks. However, frequent controller changes contradict the difficulty for verifiers to access the latest block data in an asynchronous environment. The revoked controllers may impersonate the identity and access illegal resources. To address this problem, we design a lightweight identity authentication scheme. Specifically, instead of synchronizing all heavy blocks, the verifiers only download block headers regularly. The holders are required to provide digest freshness proof, which proves that the digest is contained by the newest block.

4.4.1 Credential Issuance

The issuers can issue credentials to the holders. The credentials contain three parts, i.e., metadata, claim, and signature. Metadata contains credential properties, such as the identifier of the issuer and holder, and the unique identifier of credentials. Claims are expressed as key-value pairs, like `{age: 25}`. The signature sig_I of the issuers can guarantee the integrity of the credentials. The signature also proves that the credential is issued by authorities. This is because the adversary cannot forge the signature without the authorities' private keys. Then, the issuers send the generated credentials to the controllers.

4.4.2 Credential Presentation

To make the credentials non-transferable and resist impersonation attacks caused by revoked controllers, as shown in Fig. 4.4, the holders are required to provide digest freshness proof prv_{digest}, pk validation proof prv_{pk}, and ownership proof prv_{own}, when presenting credentials. prv_{digest} proves that the proving digest is contained by the new block. prv_{pk} proves the validation of the public key used for credential verification. prv_{own} proves the ownership over the credential.

The digest freshness proof prv_{digest} proves that the providing digest is the latest. To resist impersonation attacks, the verifiers regularly download the newest block headers rather than synchronize blocks. The controllers are required to prove that the corresponding $mapping(id, digest)$ is contained by the newest block of Ethereum. The proof can be obtained from the full nodes, like Infura.

The data structure of the Ethereum system is account-based. Fig. 4.5 plots the underlying storage of the mapping. Specifically, the Ethereum system takes advantage of three trees to keep the transaction and smart contract information, i.e., transaction tree, receipt tree, and state tree. The roots of the three trees are recorded in the block

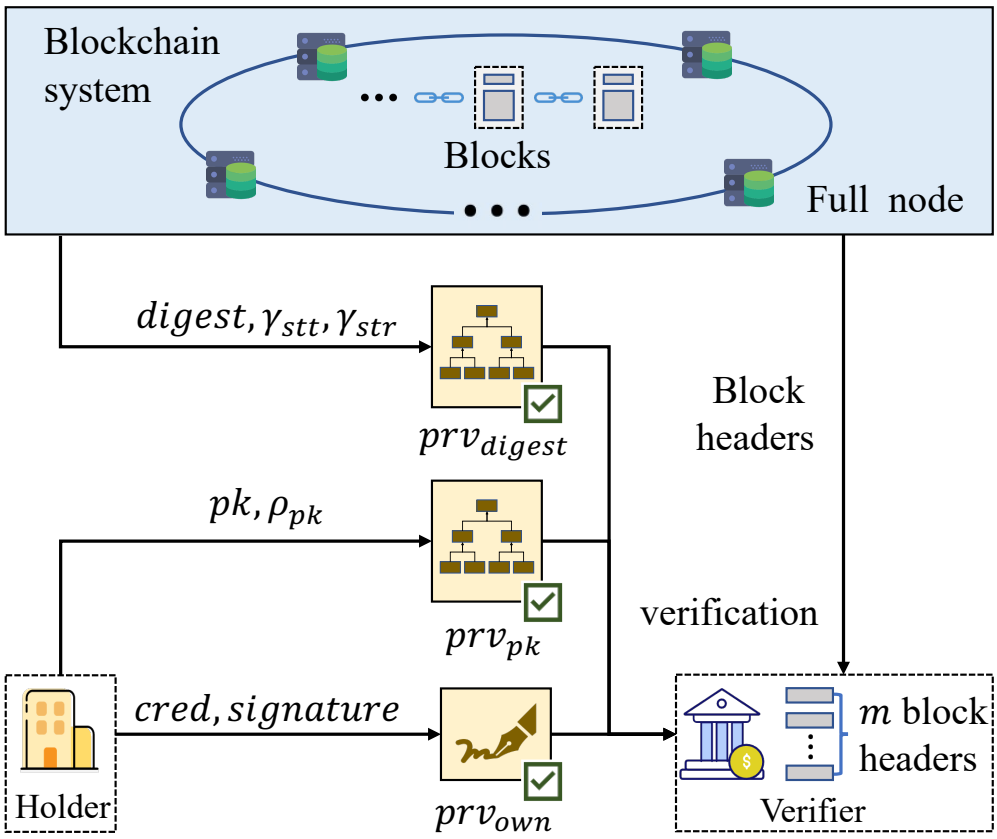


Figure 4.4: A Secure Credential Presentation and Verification with Digest Freshness.

headers. The transaction tree and receipt tree contain all the transaction information and receipt information respectively. State tree records all the accounts and their state, including the mapping between the smart contract account and its state. The $mapping(id, digest)$, as well as other data related to the smart contract, is stored in the storage tree. The root of the storage tree is maintained in the corresponding state field of the smart contract account. The account-based feature makes the traditional Merkle tree data structure difficult to apply directly in the Ethereum system. This is because frequent updates of account state may lead to high storage overhead or inconsistent status of blocks. Thus, the Ethereum system adopts Merkle Patricia Trie (MPT) [74]. MPT is a special Merkle tree that efficiently stores key-value pairs by compressing the keys. It has high retrieval efficiency and supports efficient member-

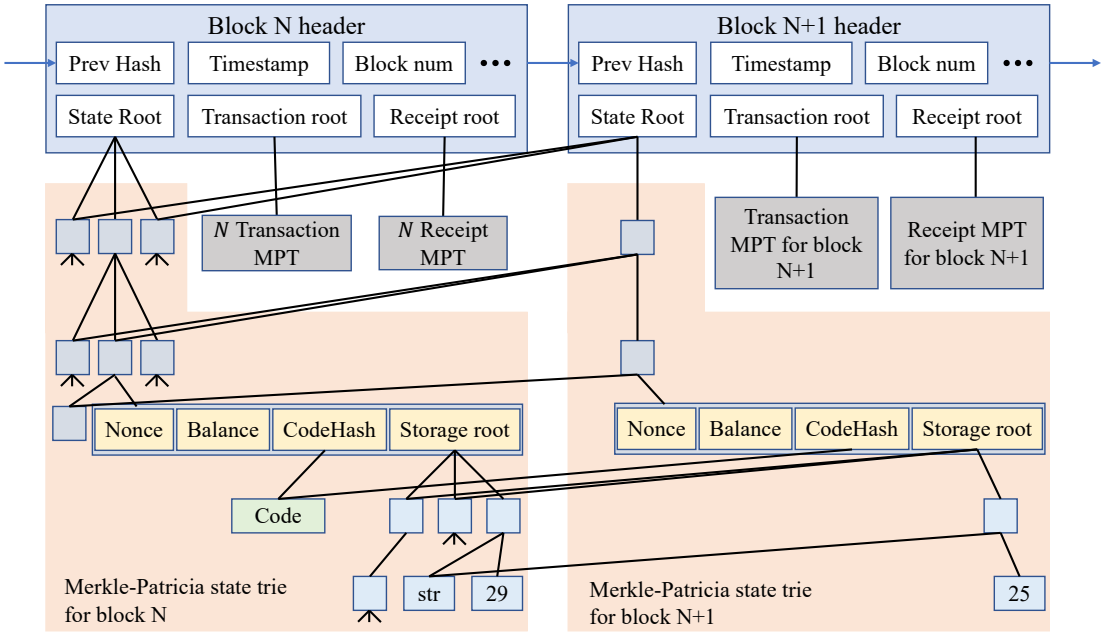


Figure 4.5: The Data Structure of the Ethereum System.

ship proof. Thus, the digest freshness proof comprises two Merkle proofs of MPT γ_{str} and γ_{stt}. γ_{str} proves that $mapping(id, digest)$ is maintained in the storage tree of the smart contract. γ_{stt} proves that the smart contract and its state are stored in the state tree. Like Merkle proof, the membership proof of MPT is the hash value of the encountered sibling nodes along the path upwards, starting from the target leaf node. With the state root recorded in the block header, the validation of the digest can be verified.

pk validation proof prv_{pk} proves that the public key is validated for credential verification. Specifically, the controllers generate the Merkle path ρ_{pk} of the latest digest. This can prove that the public key is valid since his public key is contained by the digest with good freshness.

The ownership proof prv_{own} is used to prove the ownership over the presented credentials. The above two proofs only prove the validation of the public key, but cannot prove that the person presenting the credential holds the corresponding private key. Thus, the controllers can sign for the credentials using their private key and get sig_H.

The signature can guarantee the integrity of the credentials. The signature also proves that the signer has the correct private key to control the identity. The credentials are non-transferable because only controllers with the correct private key can use the credentials.

Thus, the credential presentation (VP) is made up of verifiable credentials and proof from controllers, which can be defined as equation 4.1.

$$vp = \{metadata, claim, sig_I, \gamma_{str}, \gamma_{stt}, \rho_{pk}, sig_H\}. \tag{4.1}$$

4.4.3 Credential Verification

After receiving the presented credentials, the verifiers verify the validation of the credentials. To make the identity system resist controller-correlation attacks and identity impersonation attacks caused by insecure controller revocation, the verifiers are required to regularly download block headers from full nodes and successively verify digest freshness proof, *pk* validation proof, and ownership proof.

In our system, the verifiers only maintain several fresh block headers for credential verification. In traditional systems, the verifiers are required to synchronize all heavy blocks. They may have difficulty in promptly synchronizing the latest blocks, and thus obtain stale digest for verification, due to long block synchronization latency in an asynchronous environment. In our system, the verifiers regularly download block headers from full nodes and locally keep a certain number of block headers for verification. Block headers are accessible, because of their lower block propagation latency. The transmission latency is lower because of its small size. The propagation of block headers does not require verification, which leads to less latency.

However, in an asynchronous environment, strict freshness can reduce the system's availability. This is because the time consumption for proof generation and data transmission is not negligible, leading to unavoidable authentication failure. We make a

tradeoff of freshness and availability. The verifiers can keep a list with several block headers. The number of block headers retained by the verifiers is denoted by α. Specifically, the verifiers regularly update the block header list. If the difference between the current timestamp and the timestamp in the latest local block header is larger than the average block generation time, the verifiers modify the network configuration to get the latest global block header. This ensures that the verifiers have fresh block headers for credential verification. Our scheme substantially reduces storage overhead. In traditional systems, verifiers must store all blocks; in contrast, our approach only requires them to store a limited number of block headers. Let N represent the total number of blocks in Ethereum. According to statistics[2], the average size of Ethereum blocks is 20 KB. The average size of Ethereum block headers[3] is approximately 0.496 KB. Thus, we reduce the storage overhead for verification from $20N$ KB to 0.496α KB, where 0.496α KB is much lower than $20N$ KB.

With the state root r_s in the block headers, the verifiers verify the digest freshness proof, pk validation proof, and the ownership proof in sequence, which is defined as the equation 4.2.

$$Verify(r_s, sig_I, \gamma_{str}, \gamma_{stt}, \rho_{pk}, sig_H) \stackrel{?}{=} 1. \tag{4.2}$$

For digest freshness verification, the verifiers compute the storage root with γ_{str} and digest. Then, they verify γ_{str} with the computed storage root and the state roots in the local block header list. After verifying, the verifiers believe that the given digest is fresh enough for authentication. To verify the pk validation proof, the verifiers compute the digest by recursively computing the hash of ρ_{pk}. If the computed digest equals the proved one, the verifier considers the proved public key to be valid. Thirdly, the verifiers verify the signatures of both the controller and issuer. The signatures ensure the integrity of credentials. Besides, after the controller's signature verification, the verifiers believe the controller's ownership over the credential. After verifying the

[2]https://github.com/Ice-Storm/structure-and-interpretation-of-blockchain/blob/master/6_2.md
[3]https://blog.csdn.net/nina_1314521/article/details/130035921

issuer's signature, the verifiers believe that the credential is issued by authorities.

Security analysis. Our system achieves secure controller revocation. In our system, instead of synchronizing heavy blocks, the verifiers only download lightweight block headers regularly. The holders take responsibility for providing the freshness proof. The verifiers can verify the freshness of the digest. Thus, the revoked controllers cannot impersonate the identities.

4.5 Platform Implementation and Evaluation

In this section, we build a DID prototype platform to evaluate the performance of our system. To prove the effectiveness of our scheme, we implement the MoDID system, as the baseline. This is because in our first work, MoDID, we conduct a performance comparison with the state-of-art work. The experiments demonstrate the effectiveness of MoDID. In our second work, we mainly focus on the functional comparison with MoDID.

Firstly, we deploy our smart contract for identity management in the Sepolia Testnet. Secondly, for controller management operations, we test both time consumption for local pk Merkle tree modification and gas consumption for digest update. Thirdly, for the verifiable credential subsystem, we simulate the interactions between the credentials holders and verifiers. Specifically, we focus on parameters such as time consumption of proof generation and verification, as well as proof size.

4.5.1 Experiment Setting

We implement our system using the Ethereum blockchain. The experiments are conducted on a MacBook Air with Apple M2 CPU and 8 GB of RAM. For the controller management operations, firstly, we use Solidity 0.8.17 to compile the IdManage

smart contract. Subsequently, we deploy and test the smart contract on the Sepolia Ethereum Testnet using Truffle 5.7.2 and Infura. Additionally, for credential presentation and verification operations, the codes are written in Visual Studio Code 1.74. We use Python 3.12.0. We use the Python library Web3 6.4.0 to interact with Ethereum and use the Python library Pymerkle 6.1.0 for operations related to the Merkle tree, including building, proof generation, and verification.

In our DID ecosystem, there are $1,000$ DID subjects, including both single-controller identities and multiple-controller identities. To be compatible with MoDID, we also divided the controllers into three distinct roles, RC, delegate, and BC. We test our system using arbitrary multi-controller identity. For the identity, there are 3 RCs and 13 BCs. There are 20 credentials portraying the attributes of the DID subject. Due to the same operation process between BC and delegate management, we only choose BC as an example to test the system performance. The controllers locally maintain a pk Merkle tree containing the controllers' public keys and permission. The digest of the Merkle tree is uploaded to the blockchain.

4.5.2 Performance Analysis

Firstly, we deploy and test the IdManage[4] smart contract on the Ethereum Sepolia Testnet. The gas consumption is 1480024. The total gas fee consumption is around 0.00165 ETH.

For controller management operations, we evaluate the time consumption for local pk Merkle tree modification. Subsequently, we compared the gas consumption of our system with a baseline. Table 4.1 illustrates the gas consumption comparison alongside local time consumption. Firstly, for the time consumption, we can see that the time consumption ranges from 0.0019 ms to 0.2260 ms. The results prove that the operations related to the Merkle tree do not introduce significant time consumption.

[4]https://sepolia.etherscan.io/address/0xf2e93663F877AA4B333C2773b57c1BB33ECd8496

Table 4.1: Gas Consumption Comparison and Local Time Consumption for Controller Management Operations.

Operations	Our system		MoDID
	Time overhead	Gas overhead	Gas overhead
Create an identity	0.1600 ms	98,776	1,418,356
Add a RC	0.0458 ms	42,587	34,319
Del a RC	0.1221 ms	46,491	17,240
Add a BC	0.1130 ms	36,414	5,695
Del a BC	0.2260 ms	40,406	8,490

Table 4.2: Cost for Proof Generation and Verification.

	Proof type	Proof size (KB)	Proof generation time (ms)	Verification time (ms)
MoDID	prv_{own}	0.419	4.804	20.193
Our system	prv_{own}	0.419	5.515	6.328
	prv_{digest}	2.000	23.226	2.602
	prv_{pk}	0.337	0.011	0.010

Secondly, from the gas consumption comparison, we can see that the gas consumption of identity creation operations is much less than the baseline. Regarding RC and BC management operations, they result in slightly higher gas consumption compared to MoDID. Even though the computation of the Merkle tree leads to extra time and gas consumption, our scheme can make the identity management system reduce the risk of privacy leakage.

In the authentication process, we evaluate the cost of credential presentation and verification process. We evaluate the proof size, and time consumption involved in

these processes. Table 4.2 presents the comparison of the cost for proof generation and verification. Firstly, the comparison reveals that in our system, the proof size exceeds the baseline by 2.337 KB. This increase is attributed to the additional requirements of providing digest freshness proof (prv_{digest}) and pk validation proof (prv_{pk}) alongside the ownership proof. Despite the resultant increase in proof size, our system effectively mitigates controller-correlation and impersonation attacks. Secondly, compared with MoDID, our system leads to higher proof generation time consumption. This is because to resist identity impersonation attacks, the holders are required to prove that their digest is contained in the newest block. The digest freshness generation requires retrieving the data from the whole blocks. Thirdly, our system consumes less verification time. In our system, rather than retrieving public keys and permission from the whole blocks, the verifier only performs a series of hash operations locally for verification. The results indicate that the system has good overall performance. This is because a holder occasionally initiates authentication requests, while a verifier handles substantial verification work.

Specifically, we conduct detailed tests on the efficiency of the credential verification scheme. In the MoDID system, the verifier retrieves the controllers' public key and permission from the blockchain, and then verifies the signatures. While in our system, verifiers verify the digest freshness proof, check the controller's public key and permission, and verify the signatures. The verification process in our system does not involve time-consuming on-chain data retrieval operations. Since we do not consider credential revocation in our second work, we only compare the time consumption for credential revocation without revocation checks. Fig. 4.6 plots the variation of time consumption with the number of credentials. We can see that time consumption increases with the number of verifying credentials. Additionally, the growth rate in our system was significantly lower than the MoDID system. Thus, our credential verification scheme is low-overhead. The high verification efficiency for identity management enables our system with promising scalability.

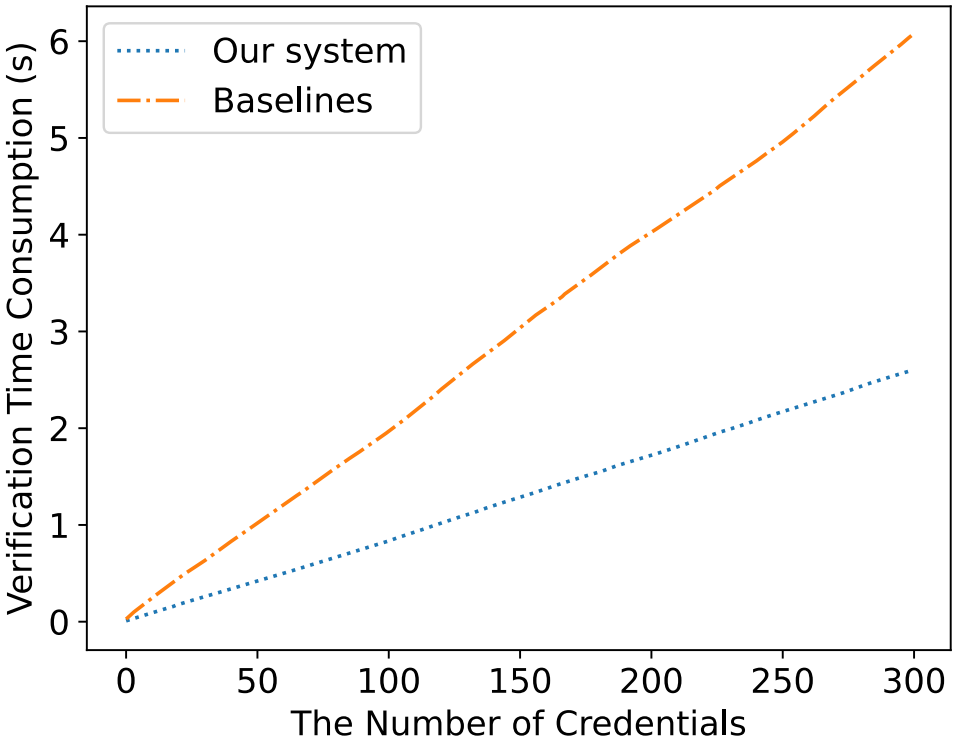


Figure 4.6: Time Consumption Comparison for Credential Verification.

4.6 Chapter Summary

In this chapter, we present a secure and privacy-preserving identity management scheme for multiple controllers, to address relationship privacy leakage and identity impersonation issues that existing DID systems have not adequately addressed. First, to address the relationship privacy leakage issues caused by the linkability of public keys, we develop a Merkle tree-based masking mechanism. This mechanism obscures identity association by transforming public keys into digests. Second, to resist identity impersonation attacks caused by high block synchronization latency, we introduce a lightweight authentication scheme. Our scheme reduces the risk of identity impersonation by liberating verifiers from the burden of synchronizing heavy blocks. Finally, comprehensive evaluations demonstrate that our scheme significantly enhances security and privacy while maintaining high throughput and low overhead,

compared with state-of-the-art work.

Chapter 5

Conclusions and Future Research

5.1 Conclusion

In this thesis, we propose a secure and privacy-preserving identity management scheme for multiple controllers. We focus on the issues caused by both existing coarse-grained identity management and the intrinsic characteristics of blockchain systems. We solve the two kinds of issues in our first and second work respectively.

- We propose MoDID to solve the problems caused by coarse-grained identity management, i.e., identity impersonation by malicious controllers and high key recovery overhead. MoDID is a fine-grained and W3C-standard compatible identity management scheme for multiple controllers. To address the identity impersonation issues in the existing DID system, we propose a hierarchical identity management system, allocating different levels of credential access and controller management permission to distinct controllers. Additionally, we propose an optimized key recovery scheme for multi-controller scenarios with less overhead than existing systems. To validate the feasibility and practicality of MoDID, we develop a prototype system. The experimental result shows that

MoDID allows multiple controllers to securely control identities while incurring less time and gas consumption than the uPort system.

- We find two new attacks caused by blockchain's characteristics, i.e., controller correlation attacks and identity impersonation attacks by revoked controllers. To resist the two problems in the existing systems, we propose a privacy-preserving and secure identity management scheme. For controller-correlation attacks, we design a privacy-preserving controller management scheme, enabling the controllers to prove their ownership over the multi-controller identities without publicizing their public keys. Additionally, to resist the identity impersonation caused by insecure controller revocation, we design a lightweight authentication scheme with secure controller revocation, freeing the verifiers from synchronizing heavy blocks to reduce the risk of identity impersonation. Finally, we implement our system on the Sepolia TestNet to evaluate the practicality of our proposed scheme. The result demonstrates that our system allows multiple controllers to control an identity in a secure and privacy-preserving manner with acceptable gas and time consumption, compared with the state-of-art work.

5.2 Future Work

Even though we propose a secure and privacy-preserving identity management scheme for multiple controllers, our current work certainly has some limitations. In future work, we will focus on further improving the security performance of the current systems for multiple controllers.

- **Controller-correlation attacks from the verifiers.** Even though we propose a privacy-preserving scheme to resist controller-correlation attacks from inferring the relationship between different identities, we assume that the verifiers are not curious about the relationship privacy between different identities.

However, it is possible for verifiers to obtain relationship information by linking different public keys and hashes of public keys in the credentials. In our scheme, the controllers expose the identifiers, public keys, and Merkle proof to the verifiers. The Merkle proofs contain the hashes of public keys. The public key and hash value correspond one-to-one. Thus, after long-time accumulation or collusion with other verifiers, the verifiers can launch controller-correlation attacks by linking the public keys and the hash of public keys. In the future, we plan to combine the zero-knowledge proof [64] and the accumulator algorithms [51], which can allow the controllers to prove ownership over the multi-controller identities even without their public keys or the hash values of public keys to the verifiers.

- **Identifier-correlation attacks.** Apart from the linkability of controllers' public keys, the linkability of the identifiers may also lead to serious privacy leakage problems. Although the users can disclose minimal information to verifiers for authentication, it is possible for the verifiers to launch identifier-correlation attacks and to access extra privacy information by concluding with other verifiers by colluding with other verifiers. Specifically, it is common for users to present credentials containing distinct privacy information to multiple verifiers. Different verifiers can correlate the received credentials by linking the identifiers recorded in the credentials. This can pose a great threat to the privacy data of users. Thus, it is important to hide the identifier in the authentication process. In the future, We can take advantage of anonymous credentials [14, 72, 44, 36] by using cryptography technologies. This can allow the users to present credentials for authentication without disclosing their identifiers. Thus, the verifiers cannot infer the privacy of the users by correlating the identifiers recorded in the credentials.

- **Sybil attacks.** Sybil attacks impede the application of DID in multi-controller scenarios. The Sybil attack means that the attacker can create multiple iden-

tities to access web resources at a relatively low cost. Many Web3 applications, such as airdrop distributions, require identity systems that are resilient to Sybil attacks. This is because a successful Sybil attack could potentially lead to huge property losses for airdrop campaigns. Existing Sybil-resistance solutions normally rely on the biometric attributes of human beings [1] or rely on the current identifier of authorities [59, 33]. These solutions cannot be applied in multi-controller scenarios directly. We plan to propose a low-overhead Sybil-resistance scheme to improve the threshold to launch Sybil attacks for multi-controller scenarios.

[1] https://zh-cn.worldcoin.org/

References

[1] Andronicus A Akinyelu. Advances in spam detection for email spam, web spam, social network spam, and review spam: Ml-based and nature-inspired-based techniques. *Journal of Computer Security*, 29(5):473–529, 2021.

[2] Frederik Armknecht, Ghassan O Karame, Avikarsha Mandal, Franck Youssef, and Erik Zenner. Ripple: Overview and outlook. In *Trust and Trustworthy Computing: 8th International Conference, TRUST 2015, Heraklion, Greece, August 24-26, 2015, Proceedings 8*, pages 163–180. Springer, 2015.

[3] Oscar Avellaneda, Alan Bachmann, Abbie Barbir, Joni Brenan, Pamela Dingle, Kim Hamilton Duffy, Eve Maler, Drummond Reed, and Manu Sporny. Decentralized identity: Where did it come from and where is it going? *IEEE Communications Standards Magazine*, 3(4):10–13, 2019.

[4] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[5] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE symposium on security and privacy*, pages 104–121. IEEE, 2015.

[6] Vitalik Buterin. Ethereum: platform review. *Opportunities and Challenges for Private and Consortium Blockchains*, 45, 2016.

[7] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.

[8] Diego Cagigas, Judith Clifton, Daniel Diaz-Fuentes, and Marcos Fernández-Gutiérrez. Blockchain for public services: A systematic literature review. *IEEE Access*, 9:13904–13921, 2021.

[9] Wei Cai, Zehua Wang, Jason B Ernst, Zhen Hong, Chen Feng, and Victor CM Leung. Decentralized applications: The blockchain-empowered software system. *IEEE access*, 6:53019–53033, 2018.

[10] Chuan Chen, Lei Zhang, Yihao Li, Tianchi Liao, Siran Zhao, Zibin Zheng, Huawei Huang, and Jiajing Wu. When digital economy meets web3. 0: Applications and challenges. *IEEE Open Journal of the Computer Society*, 3:233–245, 2022.

[11] Jan De Clercq. Single sign-on architectures. In *International Conference on Infrastructure Security*, pages 40–58. Springer, 2002.

[12] Andrea De Salve, Andrea Lisi, Paolo Mori, and Laura Ricci. Selective disclosure in self-sovereign identity based on hashed values. In *2022 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–8, 2022.

[13] Tien Tuan Anh Dinh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Ji Wang. Untangling blockchain: A data processing view of blockchain systems. *IEEE transactions on knowledge and data engineering*, 30(7):1366–1385, 2018.

[14] Changlai Du, Hexuan Yu, Yang Xiao, Y Thomas Hou, Angelos D Keromytis, and Wenjing Lou. UCBlocker: Unwanted call blocking using anonymous authentication. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 445–462, 2023.

[15] Paul Dunphy and Fabien AP Petitcolas. A first look at identity management schemes on the blockchain. *IEEE security & privacy*, 16(4):20–29, 2018.

[16] Ji Fang, Cao Yan, and Chen Yan. Centralized identity authentication research based on management application platform. In *2009 First International Conference on Information Science and Engineering*, pages 2292–2295. IEEE, 2009.

[17] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[18] Harry Halpin. Nym Credentials: Privacy-preserving Decentralized Identity with Blockchains. In *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 56–67. IEEE, 2020.

[19] Tom Heath and Enrico Motta. Ease of interaction plus ease of integration: Combining web2. 0 and the semantic web in a reviewing site. *Journal of Web Semantics*, 6(1):76–83, 2008.

[20] Friðrik Þ Hjálmarsson, Gunnlaugur K Hreiðarsson, Mohammad Hamdaqa, and Gísli Hjálmtỳsson. Blockchain-based e-voting system. In *2018 IEEE 11th international conference on cloud computing (CLOUD)*, pages 983–986. IEEE, 2018.

[21] Aaron Holmes. *533 million Facebook users' phone numbers and personal data have been leaked online*, 2021.

[22] Jun Huang, Debiao He, Mohammad S Obaidat, Pandi Vijayakumar, Min Luo, and Kim-Kwang Raymond Choo. The application of the blockchain technology in voting systems: A review. *ACM Computing Surveys (CSUR)*, 54(3):1–28, 2021.

[23] Jian Jiang, Haixin Duan, Tao Lin, Fenglin Qin, and Hong Zhang. A federated identity management system with centralized trust and unified single sign-on. In *2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)*, pages 785–789. IEEE, 2011.

[24] Minsu Kim, Sungho Lee, Chanwon Park, Jemin Lee, and Walid Saad. Ensuring data freshness for blockchain-enabled monitoring networks. *IEEE Internet of Things Journal*, 9(12):9775–9788, 2022.

[25] Yiwei Lai, Jingyi Yang, Mingzhe Liu, Yibei Li, and Shanlin Li. Web3: Exploring decentralized technologies and applications for the future of empowerment and ownership. *Blockchains*, 1(2):111–131, 2023.

[26] Laphou Lao, Xiaohai Dai, Bin Xiao, and Songtao Guo. G-PBFT: a location-based and scalable consensus protocol for iot-blockchain applications. In *2020 IEEE international parallel and distributed processing symposium (IPDPS)*, pages 664–673. IEEE, 2020.

[27] Jan Lauinger, Jens Ernstberger, Emanuel Regnath, Mohammad Hamad, and Sebastian Steinhorst. A-poa: Anonymous proof of authorization for decentralized identity management. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2021.

[28] Sungho Lee, Minsu Kim, Jemin Lee, Ruei-Hau Hsu, and Tony QS Quek. Is Blockchain Suitable for Data Freshness? An Age-of-information Perspective. *IEEE Network*, 35(2):96–103, 2021.

[29] Xiaoxu Liu, Haoye Lu, and Amiya Nayak. A spam transformer model for sms spam detection. *IEEE Access*, 9:80253–80263, 2021.

[30] Zhuotao Liu, Yangxi Xiang, Jian Shi, Peng Gao, Haoyu Wang, Xusheng Xiao, Bihan Wen, Qi Li, and Yih-Chun Hu. Make web3. 0 connected. *IEEE transactions on dependable and secure computing*, 19(5):2965–2981, 2021.

[31] Christian Lundkvist, Rouven Heck, Joel Torstensson, Zac Mitton, and Michael Sena. Uport: A platform for self-sovereign identity. *URL: https://whitepaper. uport. me/uPort_ whitepaper_ DRAFT20170221. pdf*, 2017.

[32] Kamran Mammadzada, Mubashar Iqbal, Fredrik Milani, Luciano García-Bañuelos, and Raimundas Matulevičius. Blockchain oracles: A framework for blockchain-based applications. In *Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13–18, 2020, Proceedings 18*, pages 19–34. Springer, 2020.

[33] Deepak Maram, Harjasleen Malvai, Fan Zhang, Nerla Jean-Louis, Alexander Frolov, Tyler Kell, Tyrone Lobban, Christine Moy, Ari Juels, and Andrew Miller. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1348–1366. IEEE, 2021.

[34] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[35] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 245–254, 2001.

[36] Omid Mir, Daniel Slamanig, and René Mayrhofer. Threshold delegatable anonymous credentials with controlled and fine-grained delegation. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[37] Rahma Mukta, James Martens, Hye-young Paik, Qinghua Lu, and Salil S Kanhere. Blockchain-based Verifiable Credential Sharing with Selective Disclosure. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 959–966. IEEE, 2020.

[38] Ryan R Mullins, Michael Ahearne, Son K Lam, Zachary R Hall, and Jeffrey P Boichuk. Know your customer: How salesperson perceptions of customer rela-

tionship quality form and influence account profitability. *Journal of Marketing*, 78(6):38–58, 2014.

[39] Nitin Naik and Paul Jenkins. Uport Open-source Identity Management System: An Assessment of Self-sovereign Identity and User-centric Data Platform Built on Blockchain. In *2020 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–7. IEEE, 2020.

[40] Nitin Naik and Paul Jenkins. Sovrin network for decentralized digital identity: Analysing a self-sovereign identity system based on distributed ledger technology. In *2021 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–7, 2021.

[41] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[42] Ricardo Neisse, Gary Steri, and Igor Nai-Fovino. A blockchain-based approach for data accountability and provenance tracking. In *Proceedings of the 12th international conference on availability, reliability and security*, pages 1–10, 2017.

[43] Wei Ou, Shiying Huang, Jingjing Zheng, Qionglu Zhang, Guang Zeng, and Wenbao Han. An overview on cross-chain: Mechanism, platforms, challenges and advances. *Computer Networks*, 218:109378, 2022.

[44] Cavit Ozbay and Albert Levi. Blacklisting based anonymous authentication scheme for sharing economy. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[45] Ilker Ozcelik, Sai Medury, Justin Broaddus, and Anthony Skjellum. An overview of cryptographic accumulators. *arXiv preprint arXiv:2103.04330*, 2021.

[46] Daniela Pöhn and Wolfgang Hommel. An overview of limitations and approaches in identity management. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–10, 2020.

[47] Rui Qin, Wenwen Ding, Juanjuan Li, Sangtian Guan, Ge Wang, Yuhai Ren, and Zhiyou Qu. Web3-based decentralized autonomous organizations and operations: Architectures, models, and mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(4):2073–2082, 2022.

[48] Sumit K Rana, Arun K Rana, Sanjeev K Rana, Vishnu Sharma, Umesh Kumar Lilhore, Osamah Ibrahim Khalaf, and Antonino Galletta. Decentralized model to protect digital evidence via smart contracts using layer 2 polygon blockchain. *IEEE Access*, 2023.

[49] Partha Pratim Ray. Web3: A comprehensive review on background, technologies, applications, zero-trust architectures, challenges and future directions. *Internet of Things and Cyber-Physical Systems*, 2023.

[50] Drummond Reed, Manu Sporny, Dave Longley, Christopher Allen, Ryan Grant, Markus Sabadello, and Jonathan Holt. *Decentralized Identifiers (DIDs) v1.0*. World Wide Web Consortium (W3C), 2019.

[51] Yongjun Ren, Xinyu Liu, Qiang Wu, Ling Wang, Weijian Zhang, et al. Cryptographic accumulator and its application: A survey. *Security and Communication Networks*, 2022, 2022.

[52] Hossein Rezaeighaleh and Cliff C Zou. New secure approach to backup cryptocurrency wallets. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[53] Michael Rosenberg, Jacob White, Christina Garman, and Ian Miers. zk-creds: Flexible Anonymous Credentials from zkSNARKs and Existing Identity Infrastructure. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 790–808. IEEE, 2023.

[54] Margaret Rouse. *What is Web 1.0? - Definition from Techopedia*. (2018, July 13).

[55] Ravi S Sandhu. Role-based access control. In *Advances in computers*, volume 46, pages 237–286. Elsevier, 1998.

[56] Abdurrashid Ibrahim Sanka and Ray CC Cheung. A systematic review of blockchain scalability: Issues, solutions, analysis and future research. *Journal of Network and Computer Applications*, 195:103232, 2021.

[57] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings first international conference on peer-to-peer computing*, pages 101–102. IEEE, 2001.

[58] Simon SY Shim, Geetanjali Bhalla, and Vishnu Pendyala. Federated identity management. *Computer*, 38(12):120–122, 2005.

[59] Divya Siddarth, Sergey Ivliev, Santiago Siri, and Paula Berman. Who watches the watchmen? a review of subjective approaches for sybil-resistance in proof of personhood protocols. *Frontiers in Blockchain*, 3:46, 2020.

[60] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 507–527. Springer, 2015.

[61] Rui Song, Shang Gao, Yubo Song, and Bin Xiao. zkDET: A traceable and privacy-preserving data exchange scheme based on non-fungible token and zero-knowledge. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 224–234. IEEE, 2022.

[62] Zen Soo. *China's Didi Global fined $1.2 billion for data violations.* (2022, July 21).

[63] Manu Sporny, Dave Longley, and David Chadwick. *Verifiable Credentials Data Model 2.0.* World Wide Web Consortium (W3C), 2023.

[64] Xiaoqiang Sun, F Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE network*, 35(4):198–205, 2021.

[65] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2016.

[66] Shuai Wang, Wenwen Ding, Juanjuan Li, Yong Yuan, Liwei Ouyang, and Fei-Yue Wang. Decentralized autonomous organizations: Concept, model, and applications. *IEEE Transactions on Computational Social Systems*, 6(5):870–878, 2019.

[67] Shuai Wang, Liwei Ouyang, Yong Yuan, Xiaochun Ni, Xuan Han, and Fei-Yue Wang. Blockchain-enabled smart contracts: architecture, applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):2266–2277, 2019.

[68] Yue Xiao, Peng Zhang, and Yuhong Liu. Secure and efficient multi-signature schemes for fabric: An enterprise blockchain platform. *IEEE Transactions on Information Forensics and Security*, 16:1782–1794, 2020.

[69] Yi Xiong, Shixiong Yao, and Pei Li. D2CDIM: DID-Based Decentralized Cross-Domain Identity Management with Privacy-Preservation and Sybil-Resistance. In *International Symposium on Emerging Information Security and Applications*, pages 191–208. Springer, 2022.

[70] Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper*, 2018.

[71] Dan Yamamoto, Yuji Suga, and Kazue Sako. Formalising linked-data based verifiable credentials for selective disclosure. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 52–65. IEEE, 2022.

[72] Rupeng Yang, Man Ho Au, Qiuliang Xu, and Zuoxia Yu. Decentralized blacklistable anonymous credentials with reputation. *Computers & Security*, 85:353–371, 2019.

[73] Weining Yang, Ninghui Li, Omar Chowdhury, Aiping Xiong, and Robert W Proctor. An empirical study of mnemonic sentence-based password generation strategies. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1216–1229, 2016.

[74] Cong Yue, Zhongle Xie, Meihui Zhang, Gang Chen, Beng Chin Ooi, Sheng Wang, and Xiaokui Xiao. Analysis of indexing structures for immutable data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 925–935, 2020.

[75] Dirk A Zetzsche, Douglas W Arner, and Ross P Buckley. Decentralized finance (defi). *Journal of Financial Regulation*, 6:172–203, 2020.

[76] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105:475–491, 2020.

[77] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4):352–375, 2018.