# A self-sovereign identity management scheme using smart contracts

*Jianlin* Niu*, and *Zhiyu* Ren

He'nan Province Key Laboratory of Information Security, Zhengzhou, He'nan, China

**Abstract.** The existing self-sovereign identity management schemes have some problems, such as weak availability and security risks. To solve these problems, we proposed a cross-domain self-sovereign identity management scheme using smart contracts. This scheme takes into account the entire lifecycle of identity, especially including the cross-domain use and recovery. To preserve the privacy data of users on the blockchain, we proposed a data storage method of anchoring on blockchain. Finally, we implemented this scheme using the Solidity programming language for smart contract. This scheme has been experimentally verified to be capable of maintaining the expenditure of resources under control and having good usability. Compared with other self-sovereign identity management schemes, this scheme has better performance in terms of controllability, security and portability.

## 1 Introduction

Digital identity is a set of sets that describe the characteristics and attributes of an entity. It is the representative of an individual or organizational entity in a variety of information services. Whether digital identity can be safely and effectively managed directly restricts the security of information system services.

With the continuous advancement of human social information, various information systems emerge one after another. However, digital identity is separated from service providers and identity providers, and users need to establish different identities in different information systems. With the passage of time, identity inevitably produces fragmentation, users associate multiple identities with their legitimate identities, leading to the inevitable identity fraud and data leakage phenomenon. Currently, the traditional identity management schemes in use include OpenID, SAML [1], OAuth [2], etc. However, the inherent centralized identity storage management mode of these identity management systems makes them face huge security threats. The fragmentation of identity and centralized storage bring risks such as identity theft, privacy leakage and data leakage [3]. It is urgent to propose a more secure and effective cross-domain identity management scheme than the traditional one.

Self-sovereign identity management is a method different from traditional identity management, in which individuals have the complete ownership and management right of

* Corresponding author: niu-jianlin@foxmail.com

identity. It puts more control of user identity information into the hands of users themselves, greatly reducing the risk of user identity information disclosure. With the advent of blockchain, self-sovereign identity management can be implemented step by step. Many organizations and scholars have made useful explorations on the combination of blockchain and identity management. For example, the PKIoverheid project [4], which used bitcoin blockchain in the early stage, is extremely unfriendly to users due to the slow consensus and other characteristics of bitcoin blockchain. However, the early exploration showed the huge advantages of using identity system based on blockchain over traditional solutions. Subsequently, a variety of identity management systems based on blockchain emerged, the most representative of which are uPort scheme, ShoCard scheme and Sovrin scheme [5]. The uPort scheme [6] establishes a self-sovereign identity based on the Ethereum platform, but it does not authenticate the user's uPortID, which violates the principle of KYC (Know Your Customer) in identity management, thus bringing great security risks and legal risks. ShoCard [7] provides a trusted distributed identity, but the presence of the agency as an intermediary creates uncertainty about the availability of identity. Sovrin [8] established an access distributed ledger, but its current focus on the underlying technology rather than the user, puts its availability in question.

There are still some problems with the availability and security of existing schemes, and none of them are specifically designed for cross-domain self-sovereign identity management. Based on this, we draw lessons from previous schemes and propose a self-sovereign identity management scheme using smart contract in a cross-domain environment.

## 2 Self-sovereign identity principles and blockchain

In the essay *The Path to Self-Sovereign Identity* [10], Allen has mentioned that the concept of identity management has gone through four stages of development since its birth. Allen further proposed 10 principles for the implementation of SSI (Self-Sovereign Identity). In a subsequent white paper by Sovrin [8], these 10 principles were grouped into three categories: security, controllability, and portability, as shown in Table 1.

**Table 1.** Self-sovereign identity principles.

| **Security** | **Controllability** | **Portability** |
|---|---|---|
| Protection | Existence | Interoperability |
| Persistence | Persistence | Transparency |
| Minimalization | Control | Access |
| | Consent | |

*Security:* The identity information must be kept secure. This category contains 3 principles.

- **Protection.** The rights of users must be protected.
- **Persistence.** Identities must be long-lived.
- **Minimalization.** Disclosure of claims must be minimized.

*Controllability:* The user must be in control of who can see and access their data. This category contains 4 principles.

- **Existence.** Users must have an independent existence.
- **Persistence.** This principle is also in the category *Security*.
- **Control.** Users must control their identities.
- **Consent.** Users must agree to the use of their identity.

*Portability:* The user must be able to user their identity data wherever they want and not be tied to a single provider. This category contains 3 principles.

**Interoperability.** Identities should be as widely usable as possible.

**Transparency.** Systems and algorithms must be transparent.

**Access.** Users must have access to their own data.

These principles can be well followed when building an identity management system based on blockchain. Blockchain was born in "Bitcoin: A peer-to-peer cash system" proposed by a scholar alias Satoshi Nakamoto [9]. Blockchain can also be seen as a distributed database involving multiple nodes. The blockchain records all transactions that occur on the nodes in chronological order and packages them into blocks that are stored sequentially at each node. The transaction information is organized in a Merkle tree in each block, ensuring that each transaction is tamper-proof and non-falsifiable. So, the blockchain provides **persistence**.

Smart contracts are scripts deployed on blockchains that can be executed automatically. Through smart contract, all parties can implement the agreement in full accordance with the smart contract content, which can effectively prevent cheating and denial. The reasonable design of Smart contract can ensure that the user's identity is **controllable**. The **transparency** is also provided.

## 3 Design

The cross-domain self-sovereign identity management scheme proposed in this paper is aimed at the identity management under the access control scenario. Consortium blockchain is a compromise between public blockchain and private blockchain. Consortium blockchain not only retains some features of blockchain such as tamper-proof and traceable, but also has advantages such as fast transaction speed and low transaction costs [11]. Using the consortium blockchain can greatly improve the system availability. Therefore, combined with the consortium blockchain structure, we proposed this identity management scheme.

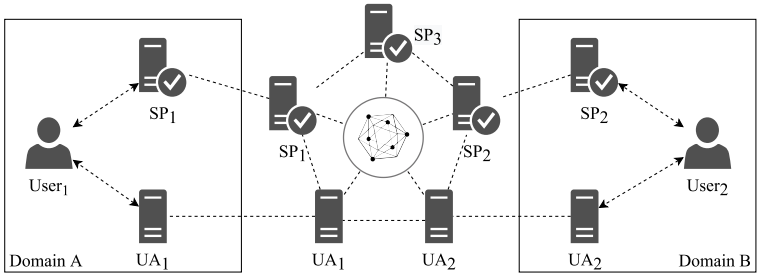


**Fig. 1.** Framework of the system based on consortium blockchain.

### 3.1 Framework

The model is based on the consortium blockchain. As the endorsement node, the SP (Service Provider) is responsible for the endorsement of this domain identity information and the authentication of cross-domain identity information. The User Agent (UA) node is also taken as the billing node. The accounting node does not participate in the consensus process, but can record the status of the current consortium blockchain, which is convenient to improve the data reading efficiency. In the consortium blockchain, cryptography

algorithms such as encryption and signature are used to ensure that each node can be trusted and the data cannot be tampered with.

In the scheme, smart contract is the core to implement identity management. We design three types of smart contract based on the principle of self-sovereign identity. The Identity Smart Contract (ISC) and Recovery Smart Contract (RSC) are deployed on the blockchain by the Service Smart Contract (SSC). Once published, the ISC is owned by the user. The SSC is initially published when the SP is placed. The ISC represents the identity of the user. Users can access the corresponding resources through their identities on these smart contracts [12]. What is shown in the Fig. 2 is the connection relationship between the smart contracts and the ownership relationship between users and the smart contracts.

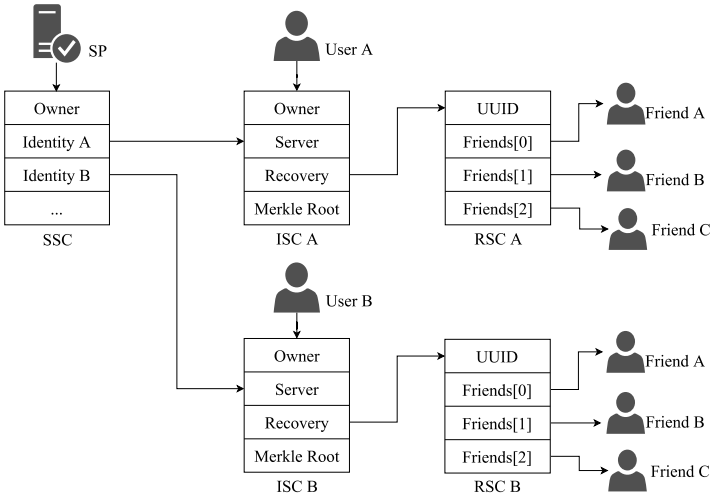


**Fig. 2.** Structure of smart contracts.

The following sections describe the functions and relationships of each smart contract.

*Service Smart Contract*. The SSC is published when the SP joins the system. After the SP has authenticated the user's identity attributes, the address of the user who submitted the request is written into the SSC. At this point, the user can publish an ISC. The SSC controls whether the user can publish an ISC to get an identity, thus ensuring the legitimacy of the identity on the blockchain.

*Identity Smart Contract*. The ISC is published by the user at the time of registration, and the ISC represents the user's identity. When the user registers his identity, the user sends a request to the SSC. If the address of the user written by SP exists in the SSC, the user is required to send information about his identity. Upon receipt of the information, the SSC publishes the user's ISC and gives control of the ISC to this user. At this point, the address of the ISC is used as the UUID (Universally Unique Identifier) of the user's identity. When an ISC is published, RSC is published at the same time and the address of the RSC is written into the data stored in the ISC.

*Recovery Smart Contract*. Since the user's identity information is controlled by himself, his public and private key pairs are at risk of being lost. Consider the availability and controllability of the identity, and join the RSC here. The RSC contains the logic to recover the user's identity. It stores the user's friends account information that can be used to recover their identity. If the identity is lost, it can be recovered with the help of his friends.

### 3.2 Registration of identity

Identity registration is the first step in using identity and the basis for ensuring identity availability. During the identity registration process, users need to publish an ISC exclusively. The address of this ISC acts as the user's identity. At the same time, the relationship with ISC and RSC is also established to help the user in the recovery of identity.

As shown in the Fig. 3, the identity registration process is as follows.

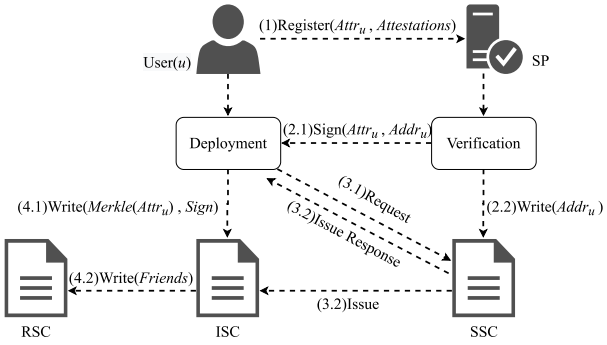


**Fig. 3.** Process of registration.

1)The user provides his/her identity attributes and identification to SP, and requests SP to endorse and verify the user.

2)After SP passes the verification, the signed information is sent to the user, and the address of the user is written into the SSC for recording.

3)The user requests the SSC to publish the ISC, and the SSC verifies whether the user address exists. If so, the user will be recorded as the user who publishes the ISC and owns the ISC. The verified property Merkle root will be written into the ISC, and the ISC address will be reported back to the user. Control is left to the user. When the ISC is created, the RSC is automatically created, and the address of the RSC is stored in the ISC.

4)The user writes the SP signature into the ISC, and then writes the friends information used to recover the identity into the RSC.

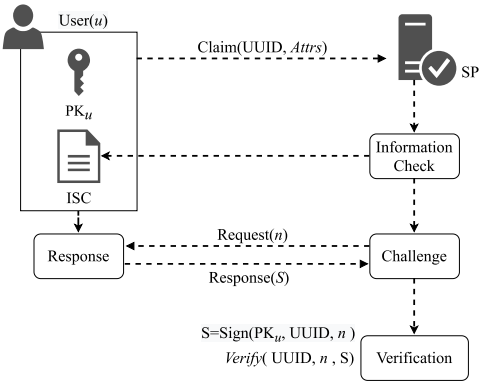


**Fig. 4.** Use of identity.

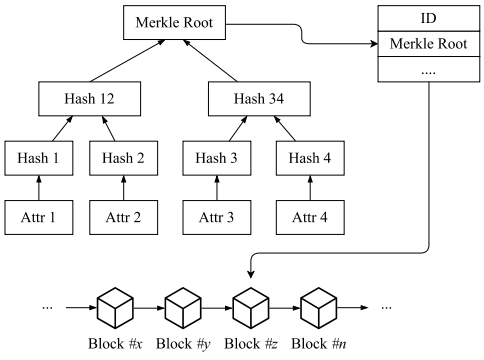


**Fig. 5.** Storage of identity attributes.

### 3.3 Use of identity

Users face two scenarios when using identity. One is the use of local identity, and the other is the use of cross-domain identity.

#### *3.3.1 Use of local identity*

As shown in Fig. 4, the using process of identity is as follows.

1)**Claim.** The user *u* declares his identity and discloses his attributes information.
2)**Information Check.** According to the attribute information disclosed by *u*, SP will check the information on the blockchain and the information recorded in ISC. If successful, it will enter the stage "Challenge-Response".
3)**Challenge.** SP selects a random number *N*, sends it to the user *u*, and requests *u* to sign it.
4)**Response.** User *u* signs *N* with his private key. The correct signature *S* can only be created if *u* has private key.
5)**Verification.** After receiving the signature *S* from user, SP will verify using the function *ResponseVerify*(UUID, m, S). If the verification is successful, then SP can offer the service to User.

#### *3.3.2 Use of cross-domain identity*

The difference between cross-domain access and local access is the disclosure of identity attributes. In local access, the collection of identity attributes is known to the SP. When accessing services provided by other SPs across domains, attributes also need to be disclosed according to the needs of specific services.

For the sake of privacy protection, the user's identity declaration (identity attribute information) should be stored on the storage device controlled by the user instead of being recorded in the blockchain [13]. However, in order to ensure the credibility and non-tampering of personal identity attribute declaration, we also need to establish a mechanism to ensure the consistency of identity information up and down the blockchain. We draw lessons from the structure block body in the blockchain, and put forward an attribute data storage method based on Merkle tree. This method ensures the consistency of identity attribute information on and off the blockchain, and at the same time satisfies the requirement of identity attribute disclosure.

As shown in the Fig. 5, in the user's device, we organize the identity attributes in the form of a Merkle tree, and the Merkle root generated by the identity attributes is stored in the ISC along with the publishment of the ISC. If the identity attribute needs to be disclosed when accessing the SP, this structure can realize the disclosure of attribute information on demand, thus avoiding unnecessary privacy disclosure.

How does the service provider verify the authenticity of the disclosed identity attribute? In the Fig. 5, we illustrate with the example of *Attr4*. When *Attr4* is disclosed, only Hash values required in the process of generating the Merkle root need to be sent together. In this case, *Hash3* and *Hash12* need to be sent at the same time. The SP generates the Merkle root based on the disclosed information and compares it with the Merkle root stored on the blockchain declared before. The detailed process of disclosure is as follows.

1) $SP \rightarrow User$: $En_{pk_U}\left(Sig_{SP}\left(Request\left(attrs\right),T_1\right),pk_{SP}\right)$

The SP sends a request to User to disclose the identity attribute attrs.

2) $User \rightarrow SP : En_{pk_{SP}}\left(attrs, T_2\right)$

User decrypts the information, if the timestamp $T_1$ is alive and verifies that SP has passed, then accepts the request, selects the property to be disclosed, and encrypts and sends it using the public key $pk_{SP}$ provided by SP.

3) $SP \leftrightarrow blockchain : Verify(attrs)$

SP decrypts the response and verifies the attributes on the blockchain, if the timestamp $T_1$ is alive.

### 3.4 Recovery of identity

Users run the risk of losing their identity, because they store personal data on their own devices. Then recovery of user's identity becomes an important means to protect identity availability.

When registering an identity, the user's friends' identity address is pre-written into the RSC. When the user needs to recover the identity, the friend sends a request to recover the identity through his own ISC. The specific process is as shown in the Fig. 6.

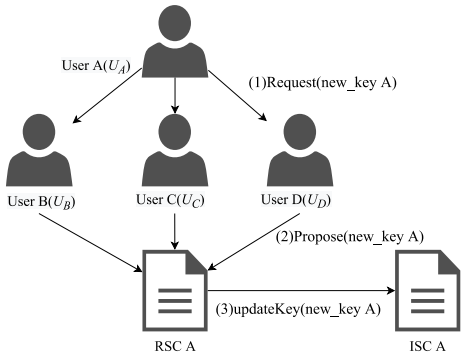


**Fig. 6.** Recovery of identity.

1)The user sends a recovery request to his friend and tells him his new public key.
2)The friends send a recovery request to the user's RSC through friends' own ISC, and sends the user's new public key.
3)After the RSC has collected enough requests, the operation of updating the public key shall be performed. At this point, the identity recovery operation is complete.

### 3.5 Revocation of identity

A revocation request can be made when the user decides to no longer use this identity. The ISC of this user is no longer available after the identity is cancelled. When an identity is cancelled, the SP will be notified that this identity will not be available to prevent the identity from being used maliciously.

## 4 Implementation

In Ethereum, Gas is the unit of measure of “effort” required to execute smart contracts. Accordingly, we can use Gas values to reflect resource consumption when deploying and executing smart contracts. This scheme has been implemented through the solidity (a smart contract programming language). The availability of this scheme was evaluated by

compiling and deploying smart contracts in a remix-IDE environment and then testing smart contracts in its built-in Javascript VM.

### 4.1 Cost

We assume that $1\,Gas = 1\,Wei(10^{-9}\ ether)$. In July 2020, the exchange rate is $1\,ether = \$228$.

As can be seen from the data in the Table 2, the resource consumption of the smart contract is relatively large in publishing, reaching the level of millions. The publishing of SSC costs the most, at $0.282, and adding an identity costs $ 0.25 (publishing ISC and RSC). The gas cost of key functional functions is less. As you can see, the gas cost of a transaction to add a friend grows linearly (by around $0.005 per friend). This result is consistent with the subjective judgment. When recovering identity, the cost of each friend's recovery proposal is basically the same under a different number of friends. The average cost of each friend's recovery proposal for a user is about $0.012. To sum up, gas costs within a reasonable range, consumes controllable amount of resources, and has good availability.

**Table 2.** Gas costs of different functions.

| Key Modules | Costs (gas) | USD ($) |
|---|---|---|
| Publishing SSC | 1238127 | 0.282 |
| Adding identity | 1098476 | 0.250 |
| Setting 3 friends | 89574 | 0.020 |
| Setting 5 friends | 131616 | 0.030 |
| Setting 7 friends | 173658 | 0.040 |
| Recovering identity (average cost of 3 friends) | 51444 | 0.012 |
| Recovering identity (average cost of 5 friends) | 53304 | 0.012 |
| Recovering identity (average cost of 7 friends) | 56852 | 0.013 |
| Setting Merkle hash root of attributes | 62780 | 0.014 |

## 5 Evaluation

In Chapter 4, we have discussed the resource consumption and availability of our solution. In this section, we will compare our scheme with others in three respects. Then, we will analyse the security of our solution.

### 5.1 Comparison of schemes

From the three principles of self-sovereign identity: security, controllability and portability, compared with other schemes, conclusions can be drawn as shown in the Table 3. As we can see, compared with other self-sovereign identity management schemes, our scheme has better performance in terms of security, controllability and portability.

**Table 3.** Comparison between our scheme and others.

| Schemes | Security | Controllability | Portability |
|---|---|---|---|
| uPort | **No.** Failure to authenticate the user's uPortID, violating the principle of KYC in identity management. | **Yes** | **Yes** |
| ShoCard | **Yes** | **No.** The role of the agency as the intermediary occurs, which makes it impossible for the user to fully control it. | **Yes** |
| Sovrin | **Yes** | **Yes** | **No.** System opacity, availability and portability to be tested. |
| Our scheme | **Yes.** User access authentication, while ensuring user privacy security. | **Yes.** Once the identity is assigned, the user has full control. | **Yes.** System is completely transparent. |

## 5.2 Security analysis

### *5.2.1 Replay attack*

In the process of disclosure and use of identity attributes, if the message is recorded and played back, it will be vulnerable to replay attack. At this time there will be the risk of identity being exploited. In this scheme, random numbers and timestamp are used in the identity disclosure process to prevent replay attacks, while the challenge-response mechanism is used in the identity use process to prevent such attacks. Therefore, the risk of replay attack is reduced in this scheme.

### *5.2.2 DDoS attack*

As the system scheme is based on the distributed architecture of blockchain, the system scheme naturally has the characteristics of no single point of failure, multiple redundancy and so on, and can resist the DDoS attack naturally. From this point of view, blockchain-based identity management has greater advantages than traditional centralized identity management.

### *5.2.3 Privacy protection*

In terms of privacy protection, the system scheme is also considered. In the system, the more sensitive identity attribute data is stored by anchoring the attribute data on and off the blockchain. The full attribute information is stored on the user device under the blockchain, while only the root value of the Merkle tree is stored in the blockchain. Given the Hash encryption function's nature, it would be nearly impossible to recover the attribute value from this Merkle root. Therefore, the system scheme protects the user identity privacy information while preventing data tampering.

## 6 Conclusion

Aiming at the disadvantages of traditional identity management schemes, this paper proposes a cross-domain self-sovereign identity management scheme through smart contract based on blockchain technology. Compared with the traditional scheme, it reduces the burden on the server side and effectively reduces the risk that the centralized server of the traditional identity management method is vulnerable to privacy disclosure. The method of anchoring identity attribute data on and off the blockchain proposed in this scheme not only protects privacy, but also realizes tamper-proof tasks. Meanwhile, compared with other self-sovereign identity management schemes, it has better performance in terms of security, controllability and portability. However, the disadvantage of this scheme is that each identity directly uses the ISC address as the UUID, which makes the UUID unreadable for humans. Our future work is to find a way to make it human readable.

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0803603 and the National Natural Science Foundation of China under Grants 61702550 and 61802436.

## References

1. SAML Specifications | SAML XMl.org. http://saml.xml.org/saml-specifications. (Accessed 28 June 2020).
2. D. Hardt. The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749 (2012).
3. L. Jean Camp & M. Eric Johnson. *The Economics of Financial and Medical Identity Theft*. (Springer-Verlag, 2012).
4. PKIoverheid | Logius. https://www.logius.nl/diensten/pkioverheid (2018).
5. P. Dunphy & F. A.P. Petitcolas. A First Look at Identity Management Schemes on the Blockchain. *IEEE Secur. Privacy* **16**, 20–29 (2018).
6. C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton & M. Sena. UPORT: A PLATFORM FOR SELF-SOVEREIGN IDENTITY. 17 (2017).
7. SITA & ShoCard. *Travel Identity of the Future - White Paper*. (2016).
8. A. Tobin & D. Reed. *The inevitable rise of self-sovereign identity*. (2016).
9. S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. (2008).
10. C. Allen. The Path to Self-Sovereign Identity. http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html (2016).
11. Z. Li *et al*. Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* **14**, 3690–3700 (2018).
12. Z. Diebold. Self-Sovereign Identity using Smart Contracts on the Ethereum Blockchain. (2017).
13. A. Muehle, A. Gruener, T. Gayvoronskaya & C. Meinel. A Survey on Essential Components of a Self-Sovereign Identity. *Comput. Sci. Rev.* **30**, 80–86 (2018).