A Blockchain based Identity Management System Considering Reputation

Zheng Zhao
Software College
Northeastern University
Shen Yang, China, 110169
zz_neu@163.com

Yuan Liu
Software College
Northeastern University
Shen Yang, China, 110169
liuyuan@swc.neu.edu.cn (Corresponding Email)

***Abstract*—In the era of the Internet, a user can perform some actions by being assigned with a pseudo identity, and most users' identities are stored by the certified third parties. This centralized identity management model has a potential safety concern. Once the database server is crashed or the data are leaked, the privacy of the user are exposed. In this paper, a smart contract technology on the blockchain is used to build a decentralized identity management system. The system is user-centric, allowing users to fully control their identity information, only authorized third parties allow access to their user information. Attribute-based authentication is used to achieve identity anonymity. In addition, a reputation model based on attribute reputation is proposed, which makes the user's identity credible even in a decentralized environment.**

Keywords—Identity, Smart contract, Blockchain, Reputation model

I. INTRODUCTION

In recent years, with the development of information technology, people are increasingly engaged in the exchange of value in the Internet. For example, people purchase daily easy-consumed products in online shopping mall frequently, and transfer money via online banking or mobile e-wallet. In China, the mobile-payment is so widely popular that people even rarely use paper money. The success of all these interactions relies on the basic requirement, that is, their online accounts is safe. Usually, users need some representative information as the network identity. These network identity information will be stored in the network of the service provider's identity information system. For example, banks access customers' information through the KYC [1] policy and store it in the banking system. Online social media (i.e., Facebook, Youtube, et al.) and e-commerce platforms (i.e., Taobao, eBay, et al.) acquire and store user identity information through user registration [5].

The centralized identity management system then becomes necessary to provide the convenience for entity users to manage their identities on the Internet. However, there are many challenging issues in these centralized systems [9]. From the perspective of online users, the existing systems bear the following shortcomings.

- Personal privacy leakage: the privately sensitive information (i.e., biological information, financial information, et al.) is often required to be included in the formation of an online virtual identity. The user privacy is then exposed to the management system, which is hard to be protected from leakage or manipulation.
- Inconvenient management: the same physical object has to repeatedly register online identities for different platforms, even though the most information provided are the same. Meanwhile, the identities of different platforms are totally isolated, which is inconvenience for users to remember and manage.

From the perspective of system management, the existing systems bear the following disadvantages.

- Low information security: the identity information is stored in a centralized server which is easy to become the attacked object of malicious competitors. The system then become dangerous and unable to ensure the normal operation of the online platforms.
- Online water army attack: the systems are hard to avoid the emergence of multiple identities from the same physical person. The multiple identities can cooperatively perform sybil attack [13], resulting in undesirable consequence for the system managers.

It is believed that identity information as users' digital assets should be properly protected. Moreover, users should be able to manage and fully control the usage of their identities. In this research, a completely decentralized identity management system based on blockchain smart contract technology is proposed. The user's identity information will be stored in the blockchain, ensuring data safety from malicious manipulation. Smart contracts are used to customize the information exchange rules based on the attributes of identities. Especially, the reputation of users is an important attribute, to be considered in our system, ensuring a credible identity management environment. In a word, an identity management system is proposed for blockchain, where the user reputation is constructed, enabling users to actively maintain a single identity in a decentralized environment.

II. RELATED WORK

A. Blockchain

Blockchain technology evolved from the Bitcoin [8]. Bitcoin is not the same as a blockchain. Bitcoin can be said to be a decentralized e-payment system using blockchain technology. After the development within the nearly ten years, the blockchain technology has also been widely recognized to be one of the most potential technologies in the coming century. At present, the blockchain has no unified

definition, and one technical definition is widely agreed as follows: a blockchain is a distributed database system composed of ordered blocks linked by a cryptographic method [3].

The data in the blockchain is stored in blocks, which are interconnected to form a chain. Each block holds the hash of its previous block, and the data changes of one block will affect all the blocks in the chain [11]. Blockchain technology uses distributed consensus mechanisms to ensure the consistency of the new blocks, and the data on the longest chain is considered to be the confirmed and safest data. In general, the blockchain has the following characteristics: decentralization, high security, hard to tamper, and so on.

The chains are classified into three types: the public chain, the coalition chain, and the private chain [14]. The public chain is represented by Bitcoin and Ethereum. The users in a public chain can freely join and leave the network at any time, and participate in the chain generation and maintenance. The coalition chain refers to a number of partners to participate in the maintenance of the blockchain system, where these nodes are selected in advance to jointly control the block generation. The private chain refers to the control power is completely managed by a private organization, and the private chain usually does not has the decentralization feature. The identity management system proposed in this paper is based on the public chain, which is safe and reliable and completely decentralized.

B. Ethereum and Smart Contract

Ethereum [6] is a public blockchain platform, which allows the developers to build blockchain-based decentralized applications. The biggest feature of Ethereum is that it firstly allows the smart contracts to be written into the blockchain. A smart contract is a digital contract, which is intrinsically a piece of computer program code that needs to be executed in a trustworthy environment. A smart contract can stipulate the participants and commitment, but the smart contract can't be changed once it is deployed in a chain. Therefore, the smart contract has high credibility.

Based on the Ethereum platform and smart contract technology, there are many studies on identity systems. For example, SCPKI [2] is a smart contract-based identity system; Smart Identity [10] uses smart contracts to create smart identities for users. The identity management system proposed in this paper is built using smart contracts.

III. SYSTEM MODEL

Our system is built based on the blockchain-based smart contract technology and previous research [15]. It has decentralized and strong security attributes. In the system, the attribute information is stored in the blockchain through a smart contract and is entirely managed by the corresponding identity. Thus, the attribute is the digital asset of the identity.

In the system model, there are two important concepts: attributes and services. We learn from the attribute-based signature authentication technology [7], the application of the identity information is decomposed into a set of attributes. For authentication, it does not have to verify all the attributes of the property collection, only need to verify the necessary attribute information that can be provided, so it provides an anonymous service for the owner of the attribute. Services are mainly divided into two types: authentication service and authorization service. An authentication service is a process of validation that verifies the identity of an attribute by verifying whether the set of attributes of an identity satisfies the conditions given by the contract through a smart contract. An authorized service is authorizers created through smart contracts and used to sign attributes. Users can grant others the right to use a particular attribute through the authorizer, or they can recycle the right after the trust is over.

A. Identity Creation

We already know that in the PKI [12] authentication system, the public key is the identity. Similarly, in the blockchain system, the public key is the user's credential. Therefore, we use the public key address as the credential for the user to manage the identity. When a user owns an Ethereum account address, he has an identity. In our system we use $Address_i$ to represent the identity of user i.

B. Attribute

When an identity is created, we use the attributes to represent the information that an identity has. Usually, an identity contains multiple attributes. Formally, an identity i consists a set of attributes: U_i An attribute in our system has a complex data type. Table I describes the data structure of an attribute and the meaning of each data field.

TABLE I. THE DATA STRUCTURE OF AN ATTRIBUTE

Fields	Data Type	Necessary
hash	bytes	Yes
value	uint	Yes
reputation	uint	Yes
grantee	mapping	Option
certificate	mapping	Option

More specially, the attribute data field can be understood as follows.

Hash---It is a summary information of an attribute, which uniquely identifies the attribute. We use the information digest of the address and the description of the attribute as the hash of the attribute so that each user has a unique and distinct attribute.

Value---The value field is used to record the value of the respective attribute. For example, if the attribute is used to describe the name of an identity, then the value field is used to record the name of the identity.

Reputation---Each attribute has a reputation value which reflects the trustworthiness of the attribute exchange. In the identity management system, the exchange of identity information between the two parties of the communication is actually an exchange of identity attributes. The exchange of attributes of different identities will affect the change of attribute reputation value. For more details, we will elaborate it in the reputation model in Section 3

Grantee---Grantee field is used to indicate an identity who wants to refer this attribute. The grantee field stores the referencer's address and reference status. Only attribute owner can change the reference status.

Certificate---Certificate field is used to store identities who want to certificate this attribute. This field records multiple signatures [4].

After an attribute is created, it is then associated with two services: authentication services and authorization service,

which are specified in the next section.

C. Attribute Authorization

In many cases, the owner of the identity grants the read right to other identities by allowing them to access the attribute values. For example, a client can allow a bank to access his name and ID information so that the bank can create an account for him. In our system model, in order to accomplish such function, the operation of attribute citation is then required. In such an operation, a reference request is needed to be launched from a referencer or caller to the smart contract. The smart contract verifies the validity of the caller's identity and sends the reference request to the owner of the attribute. The owner of the attribute can decide whether to grant the attribute to the referencer. If the owner of an attribute agree to grant the attribute to the referrer, then an attribute authorization operation is completed.

1) Authorization Application. We use $Address_{caller}$ to represent the caller, $Hash_{call}$ to represent $Address_{owner}$ the hash of the attribute he wants to reference, to the address of the attribute owner, and $Address_{owner}$ to his set of attributes. The logic of an attribute authentication application is presented as in Algorithm 1.

2) Attribute Processing. When the user receives an authorization request, he needs to decide whether to authorize the applicant. If he agrees with the authorization, he will need to call the authorization handler or else do nothing. We use $Attribute_{hash}$ for the hash of the attribute to be processed, $Attribute_{caller}$ for the applicant. The logic of an attribute authentication processing is presented in Algorithm 2.

D. Attribute Signature

A user can create his/her attributes freely, however not all the attributes can pass the verification. For example, Bob creates an attribute that represents his educational qualification. When a job seeker needs to verify his academic record, Bob needs to grant this attribute. Such attribute can't be verified as long as the educational institution is able to prove that this attribute is eligible.

Algorithm 1: Attribute Authentication Application

Input: $Address_{caller}$,$Hash_{call}$,$Address_{owner}$
Output: Attribute reference status(*true* or *false*)
if *(isValid($Address_{caller}$))* **then**
  attribute = $Attributes_{owner}[Hash_{call}]$;
  if *(attribute.hash == $Hash_{call}$)* **then**
    grantee = attribute.grantee;
    grantee[$Address_{caller}$].address = $Address_{caller}$;
    grantee[$Address_{caller}$].accpted = false;
  else
    return false;
  end
else
  return false;
end

Fig. 1. Algorithm 1

Algorithm 2: Attribute Authentication Processing

Input: $Address_{caller}$,$Attribute_{hash}$
Output: Attribute authorization processing status(*true* or *false*)
address owner = attributesMap[$Attribute_{hash}$].owner;
if *(owner == msg.sender)* **then**
  attribute = attributesMap[$Attribute_{hash}$];
  grantee = attribute.grantees[$Address_{caller}$];
  if *(grantee.addr == $Address_{caller}$)* **then**
    grantee.accepted = true;
  else
    return false;
  end
else
  return false;
end

Fig. 2. Algorithm 2

In our model, we address the above issue through the attribute signature. When an attribute requires the certified by one or more identities/parties, their signatures are necessary. Actually, there often exist multiple parties to jointly certify the same attribute, that is, multiple signatures.

The certificate field of the attribute records the signature information. When then the authenticator signed this attribute, his address will be recorded in this field. When we need to verify the signer, we only need to verify that the information in this field is correct.

We use $Address_{validator}$ to represent the validator identity, $Hash_{call}$ to represent the hash of the attribute he wants to certificate, $Attribute_{owner}$ to the address of the attribute owner, and $Attribute_{owner}$ to his set of attributes. The certification and verification of an attribute are presented in Algorithm 3 and 4.

IV. Reputation Model

In the previous section, we explained the system model. Users manage identity information by manage attributes and interact with other users through attribute authorization and attribute authentication. However, there is a problem that an entity user can create identities on the blockchain almost without any cost, resulting in many false identities. The system can easily become untrustworthy as there is no third-party agency to deal with the effects of false identities in such a decentralized environment. We expect that an entity should maintain a web identity as much as possible, and an identity should have a certain degree of credibility. Therefore, we associate reputation with identity to construct an identity reputation model based on attribute.

Algorithm 3: Attribute Certification

Input: $Address_{validator}$,$Hash_{call}$,$Address_{owner}$
Output: Attribute status(*true* or $false$)
if *(isValid($Address_{validator}$))* **then**
    attribute = $Attributes_{owner}[Hash_{call}]$;
    if *(attribute.hash == $Hash_{call}$)* **then**
        certificate = attribute.certificate ;
        certificate[$Address_{validator}$] = true;
        return true;
    else
        return false;
    end
else
    return false;
end

Fig. 3. Algorithm 3

Algorithm 3: Attribute Certification

Input: $Address_{validator}$,$Hash_{call}$,$Address_{owner}$
Output: Attribute status(*true* or $false$)
if *(isValid($Address_{validator}$))* **then**
    attribute = $Attributes_{owner}[Hash_{call}]$;
    if *(attribute.hash == $Hash_{call}$)* **then**
        certificate = attribute.certificate ;
        certificate[$Address_{validator}$] = true;
        return true;
    else
        return false;
    end
else
    return false;
end

Fig. 4. Algorithm 4

A. Reputation Value Initialization

As mentioned above, the concept of attribute reputation value(ARV) is necessary to avoid the sybil identities. We use arv_i to represent the reputation value of an attribute i. ARV changes dynamically through attribute interactions. The reputation value of all the attributes of an identity determines the reputation value of the identity. We use Rp_i to represent Rp of identity i. When the number of attributes of identity i is n. Initially, Rp_i is calculated by Equation (1).

$$Rp_i = \frac{1}{n}\sum_{i=1}^{n} arv_i \quad (1)$$

It is noted that in our system, ARV is initially set as a default value v_α for a valid attribute, and the initial reputation value is set to be 0 for an invalid one.

B. Attribute Reputation Operation

In our system, attributes are the smallest unit of interaction between identities. The ARV will be changed when certain actions happen on the attribute. There are several actions that will change the ARV.

1) Attribute modification. We do not want an attribute value is frequently modified, so every time an attribute is modified, ARV is reduced. We use v_β to denote the reduction in ARV. So v_α/v_β indicates the number of times the attribute is allowed to be modified.

2) Attribute authorization. When an identity authorizes an attribute to another identity, it indicates that the trustworthiness is constructed between them. For example, if An authorizes attribute i to B, the ARV of attribute i will increases after the successful authorization, and the Rp of identity B will also increase as a consequence. We use A_{add}^{arv} to represent the increase of ARV of attribute i and B_{add}^{rp} to show the increase of B's Rp.

3) Authorization Revocation. The trustworthiness does not always exist between two identities. It is possible that disagreements between the two identities occur during the trust period to relieve the trust in advance. When this case happens, the ARV or Rp will be reduced. For example, A early terminates his/her authentication for B on the attribute i. We use B_{min}^{θ} to denote the value of A reduced ARV for attribute i, and B_{min}^{θ} denote the reduced Rp of B.

Next we discuss the details of the impact of attribute authorization and revocation authorization on Rp. When the attribute is authorized successfully, the Rp of both participating parties will increase. In order to prevent duplication of authorization between two identities, we propose an authorization counter τ. τ identifies the number of attribute authorizations between a pair of identities, and τ increases with the number of authorizations. When the authorization is successful, the transformation of A and B is shown in Equation (2) and (3).

$$A_{add}^{arv} = Rp_B * \frac{1}{2^\tau} \quad (2)$$

$$B_{add}^{rp} = \frac{1}{n_B} * |v_i - Rp_A| * \frac{1}{2^\tau} \quad (3)$$

From Equation (2) and (3), it can be seen that the increment of ARV of A is related to the reputation of B, and the increment of reputation of B is related to the reputation of A and the ARV of attribute i. From Equation (2) and (3), we can easily deduce the conclusions shown in Equation (4).

$$\lim_{\tau\to\infty} A_{add}^{arv} = \lim_{\tau\to\infty} B_{add}^{rp} \propto \lim_{\tau\to\infty} \frac{k}{2^\tau} = 0 \quad (4)$$

From Equation (4), it can be seen that with the increase in the number of attribute authorizations by both parties, and the credit gains obtained will be less.

In the same way, we use ω to denote the number of revocation authorizations for attribute i by A and B. The ARV reduction for A and the Rp reduction for B are as follows when authorized revocation occurs.

$$A_{mib}^{arv} = Rp_B * \frac{1}{2^\omega} \quad (5)$$

$$B_{min}^{rp} = \frac{1}{n_B} * |v_i - Rp_A| * 2^\omega \quad (6)$$

Similarly, we can easily deduce the conclusions from Equation (5) and (6).

$$\lim_{\omega\to\infty} A_{min}^{arv} = \lim_{\omega\to\infty} B_{min}^{rp} \propto \lim_{\omega\to\infty} \frac{k}{2^\omega} = \infty \quad (7)$$

From Equation (7), it can be seen that as the number of authorized revocations by both parties increases, Rp will decrease more and more.

The reputation model is used to give reputation to attributes. For an identity, a high Rp indicates that the identity is recognized to be trustworthy by many people. When an identity wants to improve the Rp, he needs to be trustworthy in the interactions with different identities. The most important point is that our model solves the problem of brush reputation, which makes ARV more convincing.

V. Conclusion

This paper proposes a blockchain-based identity management system and also proposes an attribute based reputation model. The system uses smart contract management identities to achieve decentralized features. In the system, an identity maintains a series of attributes, and the identity obtains the attributes of other identities through the process of attribute authorization. The validation of an attribute is achieved through attribute certification, which is further supported by multiple signature technology. In the proposed system, identities can fully control their personal attribute asset. In addition, to better manage identities and prevent false identity attacks in a decentralized environment, we also propose an attribute-based reputation model.

Acknowledgments

This work is supported by the National Natural Science Foundation for Young Scientists of China under Grant No.61702090 and No. 61702084; National Natural Science Foundation of China under Grant No.61572123, No. 61402097 and No.61602102; the National Science Foundation for Distinguished Young Scholars of China under No. 71325002; the Natural Science Foundation of Liaoning Province of China under Grant No.20170540319, No.201602261; and the Fundamental Research Funds for the Central Universities under Grant No. N162410002, N161704001, N151708005, N161704004, N151704002.

References

[1] 2018. (01 2018). https://en.wikipedia.org/wiki/KYC

[2] Mustafa Al-Bassam. 2017. SCPKI: A Smart Contract-based PKI and Identity System. In ACM Workshop on Blockchain, Cryptocurrencies and Contracts (2017). 35–40.

[3] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. 2015. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. to appear (2015), 104–121.

[4] Jung Hee Cheon and Jeong Hyun Yi. 2007. Fast Batch Verification of Multiple Signatures. Lecture Notes in Computer Science 4450 (2007), 442–457.

[5] Marcin Dabrowski and Piotr Pacyna. 2008. Overview of Identity Management. China Communications 4 (2008), 129–142.

[6] Ethereum. 2018. A Next-Generation Smart Contract and Decentralized Application Platform. (22018). https://github.com/ethereum/wiki/wiki/White-Paper

[7] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. 2011. Attribute-Based Signatures. Springer Berlin Heidelberg.

[8] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. Consulted (2008).

[9] Lidan Shou, He Bai, Ke Chen, and Gang Chen. 2014. Supporting Privacy Protection in Personalized Web Search. IEEE Transactions on Knowledge and Data Engineering 26, 2 (2014), 453–467.

[10] Smartid. 2017. Smart Identity. (2017). https://github.com/SmartIdentity/ smartId-contracts

[11] Melanie Swan. 2015. Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc.

[12] Mary R. Thompson. 2003. Certificate-based authorization policy in a PKI environment. ACM. 566–588 pages.

[13] Jie Zhang. 2009. Promoting Honesty in Electronic Marketplaces: Combining Trust Modeling and Incentive Mechanism Design. Ph.D. Dissertation. Waterloo.

[14] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. 2016. Blockchain Challenges and Opportunities: A Survey. International Journal of Web and Grid Services (2016)

[15] Yuan Liu, Zheng Zhao, Guibing Guo, Xingwei Wang, Zhehua Tan and Shuang Wang. An Identity Management System based on Blockchain. Privacy, Security and Trust(PST) 2017, Volume:1, Pages: 44-4409