IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 26, 2019, accepted July 14, 2019, date of publication August 8, 2019, date of current version August 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2933989

EIDM: A Ethereum-Based Cloud User Identity Management Protocol

SHANGPING WANG[1], RU PEI[1], AND YALING ZHANG[2]
[1]School of Science, Xi'an University of Technology, Xi'an 710048, China
[2]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

Corresponding author: Ru Pei (peiru1226@126.com)

This work was supported by the National Natural Science Foundation of China under Grants No. 61572019, Key Research and Development Program of Shaanxi (Program No. 2019GY-028).

ABSTRACT In cloud system, user identity authentication is a key problem. If design defects persist in a cloud user identity authentication scheme, direct risks of sensitive data loss and severe information breach will be incurred. At present, the main problem of cloud user identity management system is that it relies too much on third-party services. Although some third-party-detachment schemes have been proposed in recent years, most of them still rely heavily on cloud server-centered design system. In this paper, a cloud user identity management protocol based on ethereum blockchain was proposed, followed by an establishment of a simple credit management system framework. The new protocol is an improved version of CIDM (Consolidated Identity Management) referred to as EIDM (Ethrerum-based Identity Management) protocol. In the improved protocol, JWT (JSON Web Token) in OAuth 2.0 was used to introduce smart contracts into EIDM protocol, and the credit management system was added to the system so that it can provide a credible identity authentication protocol for cloud users and service providers. The new protocol solves the problem of over-reliance on third parties in the existing identity management system solutions. In the end, an analysis on the security of the new protocol showed that the EIDM protocol proposed in this paper presents more diversified security guarantees relative to the CIDM protocol. The performance evaluation results also indicated that the new protocol demonstrates better practicability and flexibility.

INDEX TERMS Cloud computing, identity management, blockchain, reputation, smart contract.

I. INTRODUCTION

With the development of the Internet, there are more and more applications to make our life more convenient. At present, people generally use a variety of mobile phone apps, and using an app means it requires registering an account and setting a password. This phenomenon means that people have to remember a lot of passwords and accounts. Although there are many web-based tools for saving passwords, such as cookies, its security has considered. In recent years, many apps designed by large Internet companies have launched an application that can be registered into the app using an account provided by another service provider. An identity management system accompanies this phenomenon. Identity management is an integrated concept that includes policy, programming, and engineering to enable authorized resources to define the identity of users accurately and to control the use of information between them [1]. The identity management system uses an identity identifier to represent the identity, which identifies the service provider to determine whether to authorize the user to utilize the service. Identity management generally uses one of three types to manage a user's personal information: the first is a piece of information that both the user and the service provider know, such as setting a password; the second is a piece of information that the user understands and identity management. Can verify it, such as Social Security Number; the third is for the user's identity characteristics, such as the user's fingerprint, iris, etc [2]. In [3], there are three types of clouds: 1 private cloud, two public clouds, and three hybrid clouds. As the sharing concept has swept the world's dominant Internet companies, cloud product service types have also emerged. In cloud computing for high-end applications, identity management is critical. If you do not manage the identity, then cloud service providers or users will face unpredictable losses. Cloud computing is a fusion of various

The associate editor coordinating the review of this manuscript and approving it for publication was Peter Langendorfer.

technologies. Its inherent complexity is intricate, and there are all kinds of software and hardware to interact. Now the number of cloud computing users has been increasing all over the world, which requires more and more different — Identity Management Technology (IDM) to provide a secure cloud computing environment. Unlike our traditional identity management, identity management used in cloud computing uses more technologies, such as encryption technology to ensure user security, and signature technology to enable the cloud to verify the legitimacy of users. In [4], it summarizes the following traditional identity management models (IDMs). Scenario steps for traditional IDMs:(1). The user logs in to the identity management (IDM) system using the username and password; (2). User requests access to the cloud service provider (CSP) data or applications; (3). The CSP requests the token from the user; (4). The user requests the token from IDM; (5). Respectively, to IDM Token, generated CSP and the user; (6). Obtained from the user using the Token issued to IDM CSP; (7). Received from the user CSP Token and Token to get themselves IDM Make a comparison; (8). If the Token comparison is successful, the user is legal. CSP agreed to visit. Traditional identity management systems have many drawbacks. For example, it may exist that an attacker colludes with an IDM server and may intercept and analyze IDM messages when they exchanged between the user and the CSP. Theft, loss, or malicious mobile code moves into the mobile — devices to capture personal information or cooperate with other malicious attackers [4]. Since the introduction of blockchain technology, various researchers have been rushing to exploit new technologies. Due to the particularity and decentralization of the CSP's structure, this can give many applications a way to solve the problem of user trust. In particular, the identity management system, which is the first level of user contact with the application, can make good use of blockchain technology. By 2019, many blockchain-based identity management systems have designed. Several applications for identity management systems based on blockchain design has the list in [5] — for example, ID2020, ShoCard, Uport, etc. But in the scenario mentioned above, the only big for the average user to the application of the machine or real identity management systems (such as banks, schools Under the identity management system of social entities such as police stations, the guarantee of the trust of the user identity management system has not mentioned that in the application of cloud computing development, the trust problem that users receive will be more than in reality. To be complicated. The motivation of this paper is to solve the trust problem caused by users in the application background and the excessive centralization of the cloud server itself in the context of cloud computing. To answer the above questions, we use CIDM (Consolidated Identity Management)protocol, smart contract, reputation system, and other knowledge to propose an identity management system based on the Ethereum blockchain, and submit an improved CIDM protocol, and call it EIDM(Ethereum-based Identity Management). The contibutions of this paper are as follow:

1) By introducing smart contracts, our identity management protocol enables parties in the system to perform identity authentication steps without interfering with each other automatically, and the reputation system can also prevent excessive cloud service providers from existing identity management systems.
2) Our new protocol EIDM (Ethereum-based Identity Management) is an improvement to the CIDM (Consolidated Identity Management) protocol. The EIDM protocol arranges smart contracts based on the CIDM protocol and uses JWT as a token between the cloud service provider and the user.

Below we will give the related wrok in Section II,the preliminaries in Section III,and give our program model and the new protocol is based on CIDM agreement, the EIDM agreement we made in Section IV, design details of the smart contract will be given in Section V, Section VI is the performance evaluation and security analysis, the final article summarized and future work in the Section VII.

II. RELATED WORK

A. REVIEW THE IDENTITY MANAGEMENT SYSTEM

In this section, we will review it from the traditional identity management system to the current identity management system, as well as the new identity management system based on blockchain technology. It is these programs that motivate our programs. In 2014, Issa Khalil et al. proposed the CIDM (Consolidated Identity Management System) scheme [4], which provides a new IDM (Identity Management System) solution for cloud users based on traditional IDMs. Detailed steps give in the Preliminaries section. Suguna M et al. proposed an identity management program in 2017 [7], but there are still defects. First, the solution cannot prevent man-in-the-middle attacks, and the channel between the CSP and the IDM system is non-secure; and we find that the protocol of the solution is CSP-centric and does not take into account that the CSP itself may not be wholly trusted. In all identity management systems based on blockchain-based cloud computing, many researchers have given different solutions to how cloud service providers integrate with blockchains. The system in this paper uses the Cloud Data Center Representative from [8], and the literature [8] establishes a strategic framework for cloud data movement through the decentralization of Ethereum. Cloud trust. In order to connect the cloud with Ethereum, a carrier is needed. The solution is to use the representative of the cloud data as a carrier to allow cloud data to store in the Ethereum blockchain through the cloud data center. The delegate also has an Ethereum address to interact with the smart contract to achieve the desired result. Daniel Augot et al. proposed in 2017 a system for authenticating identity on the Bitcoin blockchain and user-centered [9], which suggests an identity management system built on the Bitcoin blockchain. It is a big step forward in this research direction. It uses the decentralization of Bitcoin blockchain to create a user-centric identity management system for users and service providers to share information. The OpenStack

cloud platform identity management solution [10] proposed by Yapingchi et al. in 2018 has improved the problem of too simple identity management of the platform. A patent submitted by Hanan [11] in 2018 uses modern bio-simulation technology to realize the identity management of cloud users. In [12], Alex Miller mentioned using JWT as an access token between the user and the server, but he did not specifically apply the technique to the actual cloud user identity management system, but only gave the user a login DAPP. A practical suggestion when (distributed applications). Currently, there are a variety of identity authentication protocols that serve users and clients, and they are also applications of identity management systems. But they all have different security issues. There are also several applications based on identity management systems, such as Windows CardSpace [13]. Users can set the personal information they want to save, such as the login account password of each app. It will generate a Token based on this information. These tokens can prove the identity of the user, but its defect lies in its third party. There is also PRIME [14] in Europe: allows users to obtain attributes from third parties without revealing any information about themselves. But its flaw is that it needs to rely on third parties. Once the privacy data leak, the consequences are unimaginable. There is also OpenId [15]: This is a distributed authentication protocol that helps cloud users manage their multiple digital identities to control their PII (Personal Identification Number) sharing better. But it is very vulnerable to phishing attacks. Finally, there is the well-known OAuth [16], which is also a distributed protocol to help cloud users manage their identity. In this protocol, the user accesses the resources hosted on the resource server through the Token obtained by the authentication server. This article uses a JWT (JSON Web Token) [6] in OAuth2.0 [17], which will be used as one of our Tokens to make it a credential to authenticate cloud servers and users. However, as far as the OAuth protocol is concerned, its flaws are also raised in the literature [18]: including the lack of data trust and the trust of the server. To solve the problems found in the above scheme, an identity management scheme based on the Ethereum smart contract was proposed in our project, and an application of the system was completed, which apply to the CIDM (Consolidated Identity Management) protocol. EIDM (Ethereum-based Identity Management) protocol.

B. THE BLOCKCHAIN TECHNOLOGY IN IDENTITY MANAGEMENT SYSTEM

The need for blockchain based identity management is particularly noticeable in the internet age,we have face management challenges since the dawn of the internet [5]. As we all know, there are many problems in the current identity management system. For example, the cumbersome login process, although there have been many third-party authorized login programs, and has been widely used in recent years. However, whether it is a third-party authorized login or a traditional account password login, they are faced with various network attacks. For example, we will save some login information of the user in the database. If there is a problem in the database, the consequences will be unpredictable. The emergence of blockchain technology can solve the above problems for us. First, based on its own decentralized nature, our system will not require any trusted central authority. Moreover, the data on the blockchain is tamper-proof, which also prevents some illegal activities. In addition,the users do not need to provide an account number and password, and all work can be done with the blockchain itself.

III. PRELIMINARIES

In this section, we will introduce the pre-knowledge used in some of the scenarios to understand our system better.

A. ETHEREUM AND SMART CONTRACTS

Now the smart contract we use in Ethereum is the same as the basic concept of contract in life, but the difference is that the smart contract is a small computer program and stored in the blockchain. Smart contracts can help solve many existing ones. Relying on the problems of third-party platforms, rational design of smart contracts allows us to achieve the goals they want to make, such as crowdfunding, elections, and other social activities, such as the verification of a bank statement, the development of personal software, We can all use smart contracts to achieve our goals. Blockchain technology has received extensive attention in various academic fields since it was applied in Bitcoin by Nakamoto [19] in 2008. Due to its inability to be modified and decentralized, it is also supported internally by cryptography. The security has widely used in the field of verification technology in various lines. Ethereum [20] is also known as Blockchain 2.0 technology. It has a more appropriate consensus mechanism than the previous Bitcoin and more extensive use of smart contract technology to bring blockchain technology from the last single electronic currency field. Apply to various areas. Ethereum provides us with a pleasant development environment; we can use Ethereum and smart contracts to make our contribution in multiple fields.

B. JWT(JSON WEB TOKEN)

The declaration in JWT (JSON Web Token) encode as a JSON object that acts as a payload for the JSON Web Signature (JWS) structure or a plain text of the JSON Web Encryption (JWE) structure, enabling the declaration to be digitally signed or compromised. Protection. It is a self-contained token format (general tokens are of two types: certificate type, distributed by the central authority; stand-alone type, distributed by the user and distributed). This format is a good fit for our approach. JSON Web Token (JWT) is a compact declaration representation format for space-constrained environments such as HTTP Authorization headers and URI query parameters [6]. It is a self-contained token format. The principle of JWT is that after server authentication, a JSON object generates and sent back to the user. Later, when the user communicates with the server, he must send back this JSON object. JWT works by creating a JSON object that is

sent back to the user after the server authenticated. In the future, when the user communicates with the server, they send back this JSON object. The server does not save any session (session) of the data, that is, become a stateless server, and thus relatively easy to achieve expansion. In [7], we provided with a JWT interface and a JWT.

C. KEY PRESERVATION MECHANISM IN ETHEREUM WALLET

A public-private key pair used to encrypt signed transactions. To ensure that your private key not stored in plaintext in the file (that is, anyone can read it as long as it can read it), it is essential to encrypt it with a robust symmetric algorithm. This file is mainly used to store the user's public and private key pairs. The file has the following parts: (1). Cipher: is a symmetric encryption algorithm used to encrypt the Ethereum private key; (2). Cipherparams: is a public parameter used in symmetric encryption; (3). Ciphertext: dense Text; (4). kdf: is a key generation function; (5). Kdfparams: Parameters required by the key generation function [22]. Square Ethernet-based developer chooses protection password; you only need to enter a password to retrieve the decryption key. After the decryption key generated, the key is processed in conjunction with the ciphertext and compared to the MAC. If the result is the same as the MAC, the password is correct.

D. REPUTATION SYSTEM

Our system based on Ethereum Smart Contract Identity Management has a secure reputation system for CSP (Cloud Service Provider) and users to record reputation values. Just entering the system, if it is a legitimate user and a legitimate CSP, you can get the default reputation initial value. When the system initializes, the user and the cloud service provider will first specify that both the reputation values are α; in our, Ethereum based In the identity management system of the blockchain, users, and CSPs that enter the system after registering the Ethereum wallet are legal by default. However, if there is a behavior in the system that provides a false token or causes the system to operate abnormally, the reputation value will be deducted. If the operation is regular, the reputation value will not deduct.

E. CIDM

The CIDM protocol [4] proposed by Issa Khalil et al. in 2014 has attracted the attention of many researchers in this field. In Figure1, the protocol makes a new IDMs (identity management system) scheme for mobile users based on the traditional IDM protocol.

The specific steps of the protocol in this scheme are as follows:

1) The user generates a key K and a session submission value M (including the user's identity information, CSP information, IDM information, and a random value). Then use K to encrypt this M to generate $C = E(K, M)$;

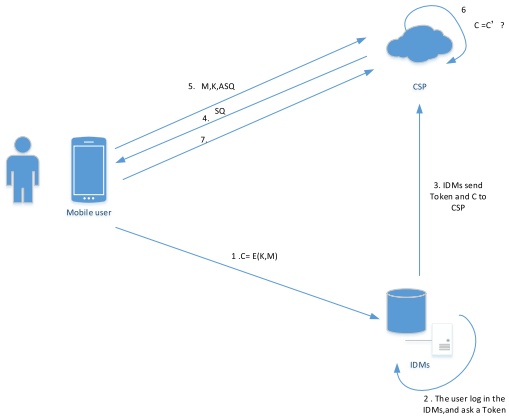

FIGURE 1. CIDM.

2) The user logs in to the IDMs (identity management system) using the account registered in the system. The user then sends C to the IDMs and requests a token from the IDMs for the following CSP to authenticate the account.
3) IDMs generate a token and send the Token and C together to the CSP;
4) CSP asks the user about the security question (SQ) (this security question can be a non-public personal question such as asking your nickname, your primary school name, etc.);
5) The user sends the M, K and security question answers (ASQ) encrypted with the CSP public key and sent to the CSP;
6) CSP decrypts with the private key to obtain the response of ASQ and M, K. Now CSP uses K to encrypt M to get C', verify and compare C sent from IDMs to see if C is equal to C';
7) If they are similar, the user can get the service of the CSP.

Compared with the traditional IDM protocol, the scheme has specific improvements. For example, he can prevent duplicate attacks, and in terms of the amount of data, it needs less data exchange, which can help users save money. However, it still has many flaws. For example, in [7], it is pointed out that the scheme cannot guarantee man-in-the-middle attacks. And when dealing with the transmission of the Security Question Answer (ASQ), the user directly transmits the answer to the security question (ASQ) to the CSP (Cloud Service Provider). If the answer to the security question is not encrypted, the transmission in plaintext must be It is not safe. Finally, in this scenario, almost all verification work is done by CSP, utterly dependent on the centralized CSP, and there is no structure similar to the reputation system to supervise its work. A wholly trusted CSP does not exist. In response to this problem, the EIDM protocol in our system is a new protocol based on the above protocol. We have retained a better design in the CIDM protocol and implemented decentralized identity authentication based on blockchain technology. The system,

coupled with smart contracts and reputation systems, adds flexibility and usability to our new protocols without losing the security of previous protocols.

IV. SYSTEM MODEL

The program consists of the following entities: users, cloud service providers, cloud data center representatives, an Ethereum wallet, two smart contracts, and an Ethereum blockchain. User: The user will register on the Ethereum wallet, and then the user can deploy the smart contract with the wallet address. The user can enter the information into the Ethereum blockchain through the agreement and use the smart contract to complete the identity authentication. The cloud contains two roles: Cloud Service Provider (CSP) and Cloud Data Center Representative (CDC) [7]. The functions of these two roles briefly explained below. Cloud Service Provider (CSP): The cloud service provider interacts with the user to further determine the user's security issues by identifying the token. Cloud Data Center Representative (CDC): The cloud data center represents the smart contract deployment of the cloud service provider and the interaction with the cloud service provider for verification. Together with verifying the legitimacy of the user's identity, in addition to the tokens in the system (JSON Web Token), needs to be generated by the private key signature represented by the cloud data center. At the same time, it needs to deploy a user's reputation value record on the smart contract to interact with the smart contract implemented by the user to form a reputation system belonging to the user for supervision. Ethereum Wallet: Can be any of the popular Ethereum wallet applications, used to create Ethereum accounts for users and cloud data center representatives, and provide public and private key pairs, mainly to ensure the security of the wallet. Smart Contracts: Smart contracts are an essential part of our system. The system deployed a total of three smart contracts, one implemented by the user and two deployed by the cloud data center. Our smart contract mainly stores the user's encrypted token and the security of the cloud server. We will use the smart contract to verify the above information and form a reputation system to make our solution more than before. Fair. Ethereum blockchain: Provide a credible environment for our solutions, so that the user's reputation value publish in the chain, which helps cloud service users to simplify the cumbersome authentication process in the future. Our system architecture, as shown in Figure 2:

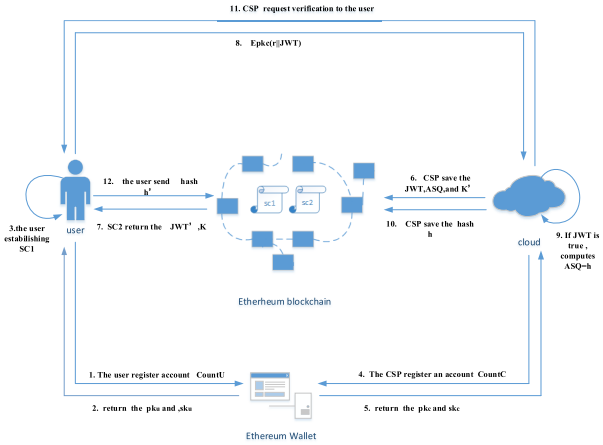

FIGURE 2. The archeticture of EIDM model.

The framework of the EIDM system solution is as follows:

1) User U registers an account on the Ethereum wallet as Count U(EOA: External Owner Account);
2) The Ethereum wallet returns a key pair to the user and records it as pk_u and sk_u,the key pair here is mainly used by user to manage his wallet account and not use elsewhere;
3) System initialization, user and cloud service provider must first specify a consistent reputation value α; the user establishes a smart contract (SC1) and enters the user's necessary information, such as the login name, etc., the login name will be in the initialized system. Then the cloud service provider can generate JWT (JSON Web Token), and the initial value of the reputation, and finally need to create a random value r, which will be used later (to be used to prevent duplicate attacks);
4) The cloud data center representative (cloud data center representative) register as the Count C on the Ethereum wallet;
5) The Ethereum wallet returns to the cloud data center to represent a key pair, and record as pk_c and sk_c, the key pair here is used by the cloud data center to manage its wallet account, and will not use elsewhere;
6) The cloud data center represents the design of the smart contract SC2 contract, and then the cloud service provider enters the cloud data center token JWT (JSON Web Token) and a security question. This security question is used to ask the user's JWT to attach the r,and generate the hash $h = (\text{JWT}\|r)$. To ensure that JWT does not leak in the Ethereum blockchain, we will use JWT (JSON Web Token) randomly selected in the AES algorithm, and use the base64 to process the encrypted JWT and record it as JWT'. Finally, the ciphertext K' is obtained by using the user's public key PK_u to encrypt the key K, and this K' is stored on the SC2 together with the encrypted JWT.
7) The cloud service provider invokes the SC2 contract, which returns the encrypted JWT' and K' to the user.
8) The user obtains K' and JWT',and decrypts its using his private key SK_u,the user obtains K and decrypted JWT, then uses the public key PK_c to encrypt r and JWT, and sends it to the CSP together;
9) The cloud service provider (CSP) decrypts the encrypted JWT received from the user with the private key, compares it with the original JWT, calls the compareJwts() method, and if the method returns *true*,then set the hash $h = (\text{JWT}\|r)$ to the answer to the security question;

10) The CSP enters the hash value into the SC2 through the cloud center data representative;
11) The CSP initiates a verification request for the security question to the user and informs the user of the address of the smart contract SC2 so that we can perform the following operations;
12) The user hashes with the encrypted JWT and r to get a hash value h', and enters the value into SC1; then user calls the smart contract SC1 to obtain the address of the smart contract SC2, and the user can read the hash value h in SC2. Enter it into the compareASQ() method and compare it with h. This method will also update the reputation value (if the error will cause the original reputation value to decrease if the correct reputation value will be added), then the smart contract SC2 will The verification results are displayed in the Ethereum blockchain.

Now we combine the specific steps in the above scheme to give us the improved CIDM protocol we used last. The Ethereum-based Identity Management (EIDM) protocol in our system solution is as follows:

1) The user first registers in the Ethereum wallet, Ethereum wallet returns the user's pair of public and private key pairs and records it as pk_u and sk_u. Then, after logging in to the Ethereum wallet, the user deploys a smart contract SC1 and saves the revealable necessary personal information and the initialized reputation system in the form of publication on the smart contract SC1. Finally, the user has to generate a random value r for future use (the random value cannot be entered in the smart contract) (corresponding to steps 1, 2, 3);
2) The cloud data center representative(CDC) will register in the Ethereum wallet, and the Ethereum wallet will return a pair of key pairs and record it as pk_c and sk_c. Then the CDC deploy a smart contract SC2, and collect user information to form a JWT (the specific JWT form will give in Section V). The JWT is encrypted with AES, and the CDC use the key K encode the JWT,and it can obtain $C_1(C_1 = \mathrm{E}(K,\mathrm{JWT}))$, and use Base64 deal with the C_1 to obtian the $C_2(C_2 = \mathrm{Base64}(C_1))$. After the user's public key PK_u is used for CDC encrypt the key K, obtain the $K'(K' = \mathrm{E}(PK_u,K))$ Finally,CDC publishes the K' and C_2 on the SC2 (corresponding to step 4, 5, 6 of the scheme);
3) After the above information deploys on the Ethereum blockchain, SC2 returns C_2 to the user and the CSP separately, and returns K' to the user; the user decrypts the K' using his private key SK_u,and obtain the K_1, then use Base64 obtain the $C_1(C_1\text{'} = \mathrm{Base64}(C_2))$. Then the user using K_1 decode the C_1 and obtain JWT'. Finally, it is encrypt r and JWT',obtain $C_3 = (PK_c, JWT'\|r)$ by the public key PK_c of the cloud service provider, and user sent C_3 to the CSP (corresponding to steps 7, 8, 9 of the scheme);
4) The CSP decrypts the C_3 with the private keyy SK_c. Now compare the previous JWT' with the previous JWT. If it is not equal, close the following steps. If the JWT is correct, add a random value r' after the JWT, then hash it to get the hash $h(h = h(JWT'\|r'))$. Finally, let this h As the future ASQ (the answer to the security question), (corresponding to step 10 of the plan);
5) The CDC (cloud data center representative) get h and write into the comparehashs() interface of the smart contract SC2. The CSP will send a verification request for the security question to the user, and then the user will input JWT' and r,and generated $h'(h' = \mathrm{h}(JWT'\|r))$ and write h' into the compareASQ() method in SC1, and SC1 will interact with SC2 to read the hash value, and then write the h in the interface of compareASQ(),then user call the method. If $h' = h$,the user's reputation value will increase. Otherwise, it reduces. The equality of hash values also indicates that the user passed the test for security issues. (corresponding to steps 11, 12 of the scheme)
6) If the above verification gives, the user and the CSP are allowed to interact (that is, the two can access each other).

V. SMART CONTRACT AND JWT DESIGN

A. SMART CONTRACT DETAILS

In this part, we give the relationship between the smart contracts involved in the scheme and the specific design and function of the functions in each contract and finally provide some algorithm logic. In Ethereum, each contract will have an address that allows us to view and access data in the contract. The deployment contract and design of each operator will also cost the corresponding Gas to reduce the cost and load of the blockchain. We also need to design efficient smart contracts. First, we give the relationship between the contracts, as shown in Figure 3: The user deploys Smart Contract SC1, which deploys Smart Contract SC2, where SC1 is implemented using the user's Ethereum wallet address, and SC2 is deployed separately with CSP's Ethereum wallet address and is publicly visible on the Ethereum blockchain. SC1 and SC2 also need to interact. SC1 needs to read some data input in SC2 on the Ethereum blockchain, which is convenient for us to use later. In the following, we will give a detailed description of the specific functions of the methods in each contract and the interpretation of some parameters.

1) The SC1 deployed by the user called the Users contract; Its primary function is first to store the user's personal information, and then to give the user an initial reputation system (including the initial value of the reputation); finally, you need to design an interface for the previous ciphertext C1 and a verification function. The most important functions of the contract is as follows: First, define a structure User to store the user's necessary information, including the user name and other information equivalent to the payload in the JWT, and the user's reputation value. In this system, the user's reputation value α just entered is defaulted (to achieve

Contract 1 Users Contract

```
pragma solidity ∧0.4.19;
contract Users{
  struct User{
  }
  address owner;
  uint public Rpt=60;
  uint numUsers;
  mapping(uint⇒ User) users;
  mapping(uint ⇒ bytes) asqs;
  modifier ownerOnly(){
       require (msg.sender == owner);
      _;
  event Userinfo(uint userID,bytes32 myASQ,
   address owner,uint Rpt);
  function newUser(string name,uint iat,uint index,
   bytes32 ASQ,bytes32 myASQ)
   public returns (uint userID){
  function compareASQ(bytes a,bytes b)
   public returns(bool)
```

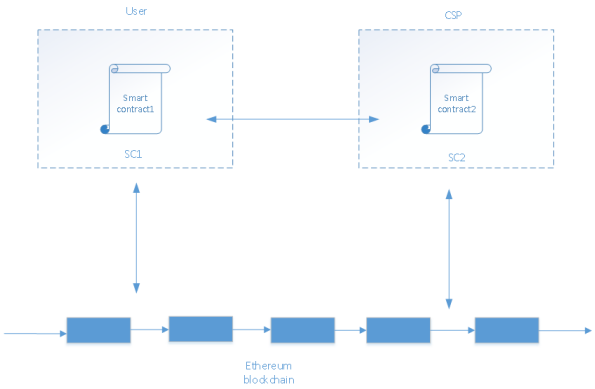

FIGURE 3. The relationship between contracts.

the contract) We set α is 60 at here, this value can change according to the specific situation), after the compareASQ() method is executed, the value will change. If the two ASQs are not equal, the reputation value will decrease, we will put the value Published in Ethereum, through the event Userinfo () we can view the user index UserID, the user's wallet address and other information corresponding to the value on the corresponding Ethereum website. Use the modifier onlyowner() to restrict the contract to only be used by one of the user's wallet addresses. In the compareASQ() method, since we can't directly compare two bytes, we use the keccak() method in solidity to calculate the hash value of the two to achieve the comparison of the two values. The UserRPT() method can return the current user's reputation value. Finally, the kill() method (This function is not given in this article,but you can see the detailed code in my github.[1])can be used by the user to destroy the contract in person. This method restricts the wallet address of the deployment contract to execute the contract.

2) SC2 deployed by CSP called Csp contract; The main function of this contract is that the user token storage (encrypted) is required first; the establishment and verification of the security problem; the second is the user's reputation value determination condition; the need to store K', $C2$, and return The user reputation value and the security question answer (ASQ) interface. The specific form of the contract is as follows: The reputation value α of the cloud service provider (to achieve the contract, we set the α contract to 60, the value can also change according to the specific situation). The Csps contract and the previously deployed Users contract have the same essential functions, and the CSP structure is different. We need to store K' which is encrypted with RSA by RSA. We record K' as keys, and record the encrypted JWT as $enJwt$. Also, since we consider more than one CSP, We give its index value $index$, as well as the hash value and the reputation value Rpt. As mentioned earlier, to minimize the cost, we can process the encrypted ciphertext to 32 bytes, which is convenient for us to implement the latter methods. The primary purpose of the modifier and kill() is also to make the contract available only to the CSP and only to use the Ethereum wallet that deploys the contract.

B. THE DESIGN OF JWT

The form of JWT is as follows:

In the official document of JWT (JSON Web Token), there is an introduction to the above information. Here we mainly explain the signature field. The meaning of the HMACSHA256 brackets means that they are base64 encoded in the base64url of the package in Node.js, and then on the opposite side. Two fields are HMACSHA256 signature algorithm with the secret. This secret place above the private key that Ethereum returns to the CSP (cloud service provider) (since the HMACSHA256 algorithm use for this information, we will not worry about it leaking). Note that to prevent third-party interception of the access token when transmitting the access token, our JWT sent in encrypted form in the system. Finally, we can use the JWT encryption format formed on the JWT official website [21] as an experiment in our later period.

VI. PERFORMANCE AND SECURTY ANALYSIS

A. ENVIRONMENTAL CONFIGURATION

For the base hardware,we use a HP Pro 3380 MT 32-bit Linux OS,Inter Core i5-3470@3.2GZ×4,4GB Memory with a AMD CEDAR GPU.Our test templates built on the Alibaba Cloud lightweight application server [24]. Fig 4 is a simply and test login page on my phone.

[1]https://github.com/flower-pp/ethereum-IDM

Contract 2 Csps Contract

```
pragma solidity ∧0.4.19;
contract Csps{
  struct CSP{
  uint index;
  uint myRpt;
  bytes keys;
  bytes enJwt;
  bytes myhash;
  }
  address owner;
  uint numCsps;
  uint cspsRpt = 60;
  mapping(uint ⇒CSP)csps;
  modifier ownerOnly(){
       require (msg.sender == owner);
      _;
  }
  event eventinfo(uint cspID,bytes32 keys,
   bytes enJwt,bytes myhashs,uint Rpt);
  function newCsp(uint index,uint myRpt,bytes keys,
   bytes enJwt,bytes myhash)
   public returns (uint cspID){
     cspID=numCsps++;
     csps[cspID] = CSP(index,myRpt,keys,
     enJwt,myhash);
  }
  function comparehashs(uint cspID,bytes a,bytes b)
   public returns(bool){
     CSP storage f = csps[cspID]
     if(keccak256(a)= keccak(b)){
       f.myRpt++
      return true
     }
     else{
       f.myRpt−−
      return false;
     }
  }
  }
```

FIGURE 4. A test web in my smart phone.

Table 1 shows the software used in our experiments and their corresponding version numbers and usages:

In the experiment, we assume that an existing user and a CSP (cloud service provider) are involved in our system,

JWT

```
Header:{
  "alg" : "HS256",
  "type" : "jwt"
} Playoad:{
  "iss" : "csp",
  "userID" : "0",
  "name" : "pp",
  "iat" : 1545778800,//unix timestamp
  "exp" : 1545951600//unix timestamp
} Signature:{
   HACSHA256(base64url
  (header)+"."+ bse64url(payload),secret)
}
```

TABLE 1. Software and version.

Software	Version	Use
Ubuntu	16.04LTS	OS
Metamask	6.0.1	Ethereum wallet
Remix	0.7.5	Solidity IDE
OpenSSL	1.0.2p	encrypt tool
Node.js	6.16.0	The back-end language
Web3.js[25]	1.0.0	API

but in practice, our system also satisfies multiple users and multiple CSPs participating in our system. Each user and CSP only needs to register an Ethereum wallet, and only the CSP's private key needs to used in our system. The OpenSSL library provides us with AES and RSA APIs. We can use Ubuntu's terminal (Shell command line) to encrypt and decrypt data. Our AES here selects the CBC mode; the key is 32 bytes, the actual size of the ciphertext after encrypting JWT (JSON Web Token) is 128 bytes; the private key length of RSA is 512 bytes; JWT is on the official website [21] generated. The actual size is 177 bytes (the final JWT form will give in the appendix); the user's random value is 4 bytes. To make the cost as small as possible, the more extensive data in the smart contract is processed and reduced to 32 bytes, and the ciphertext is converted by base64, and then the existing tool is used to add the prefix 0x to Bytes. In the form so that we can store this data on a smart contract. Finally, our contracts deploy on the Ethereum test site Rinkeby [26], and the specific contract address will give in our appendix.

B. THE COST OF OUR TEST

Two smart contracts used in our system.In addition,we give the current Ethereum capitalization is 1 ETH = $219.02. [2] The user deploys the first contract User. It costs 477901 gas (0.000478ETH), which is 0.07USD. The cost of other functions shown in Table 2.

[2]https://etherscan.io/chart/etherprice

TABLE 2. The cost for user.

Function	Used Gas(wei)	Used Ether	Used USD
newUser()	167047	0.000167	0.02
compareASQ()	30684	0.000031	0.00
UserRPT()	27218	0.000027	0.00
kill()	21700	0.000022	0.00

TABLE 3. The cost for CSP.

Function	Used Gas(wei)	Used Ether	Used USD
newCSP()	129354	0.000129	0.02
comparehashs()	30960	0.000031	0.00
kill()	21678	0.000022	0.00

In Table 2, we can see that the newUser() method consumes a total of 167047 gas, which is equivalent to one-third of the deployment contract, but the following compareASQ(), UserRPT() and kill() methods only cost a small amount of gas., which are 30684 gas, 27218 gas, and 21700 gas, respectively, converted to 0.000031 ETH, 0.000027 ETH, and 0.000022 ETH, respectively. Next, the second contract CSPs deployed by the cloud data center representatives. They respectively cost 462729 gas to synthesize ETH to 0.00463, and ETH to USD is 0.07 USD. In Table 3, we can know the cost of each method in the two contracts, newCsp() consumes 129354 gas (0.000129 ETH), converted to USD 0.02 USD, which is equivalent to one-third of the deployment contract, the latter two The methods compearehashs() and kill() also cost only 30960 gas (0.000031 ETH) and 21678 gas (0.000022 ETH), respectively.

C. PERFORMANCE COMPARISON

Table 4 gives a comparison of the data exchange between the previous CIDM protocol and our new protocol. The data exchange volume of CIDM in Table 4 comes from the literature [4]. Finally, our new protocol allows users to interact with CSPs with less data interaction. In EIDM, users need to enter information in our system. The two hash values entered by the user are 64 bytes, plus the reputation value, the ID and time are 70 bytes, and the received data size is 64 bytes, so the total It requires 134 bytes of data exchange, which is 10 bytes less than the previous CIDM. The data exchange volume of the EIDM system is 101 bytes, and finally, the data exchange volume of the cloud service provider is 162 bytes. The EIDM protocol replaces the complex program implementation with a smart contract, and it also shows its superiority in terms of data exchange, which can further save data traffic for users.

In our EIDM protocol, since the security of the user and CSP key is based on the security of the Ethereum wallet itself, in the aforementioned internal mechanism of the wallet, we can also prove that it is against man-in-the-middle attacks, as well as other attacks.. Now our main concern is the possible

TABLE 4. Performance comparison results between CIDM and EIDM.

	CIDM			OUR EIDM		
	user	cidm	csp	user	eidm	csp
Send(bytes)	104	66	40	70	37	130
Receive(bytes)	70	37	130	64	64	32
Total(bytes)	**174**	**103**	**170**	**134**	**101**	**162**

TABLE 5. Summary of experimental results.

	IDM	CIDM	OUR EIDM
IDM server compromise	Yes	No	No
Smartphone compromise	Yes	No	-
Network interception	Yes	No	No
Rely on IDM server	Yes	Yes	No
Rely on User	Yes	Yes	No

attacks within our system. In [4], we consider the external IDM server conpromise. In our EIDM system, the IDM server is the Ethereum blockchain itself. Suppose we put a wrong code into a smart contract, and the contract will detect the error at runtime. Moreover, even if you put a malicious user code, because of the reputation system of our system settings, if the user will affect its credibility, this behavior is not worth the loss. Similarly, if a CSP also puts malicious code into it, it will affect its own credibility. In the case of Newtwork traffic interception, since all of our information is encrypted in transmission, even if a malicious intermediary gets the data, the real data cannot be obtained. Then there is the case of Smart compromise. Since the mobile browser does not support the Ethereum wallet plug-in, the work on the mobile side is still unknown. However, we have already done a test interface, you can also see this interface in the phone. Then there is the situation of over-reliance on the IDM server system. Since our system is based on the blockchain, its own decentralization means that our system does not rely too much on the IDM server. Similarly, it does not rely too much on users. of. This guarantees the rights of users and protects the rights of CSP. In Table 5, we summarize the above and the comparison between our scheme and the CIDM scheme and the traditional IDMs scheme.At last,we give the cost of time in Fig 5.

D. SECURITY ANAYSIS

1) **Anonymity and privacy:** First of all, our users store virtual identities in our system, so there is no need to worry about the leakage of the user's true identity. Furthermore, the tokens, keys, and other information in our system are encrypted and transmitted, and RSA

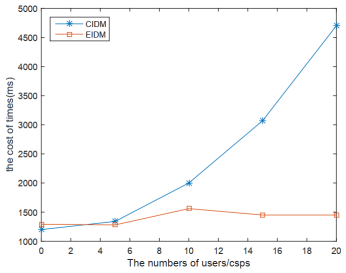

FIGURE 5. The cost of time and compare with CIDM.

and AES are used to protect users and cloud service providers from providing sufficient privacy.

2) **Preventing replay attacks:** When designing the CIDM protocol, the designer has devised a way to avoid replay attacks by letting the user take a random number r so that the same user cannot use the same random number to enter our system. The random number will change, and this measure can well protect against replay attacks.
3) **Prevent users from collusion attacks:** Assuming that User A and User B collude to attack our system, CSP (Cloud Service Provider) still sends them the public key PK_c and K', but since they do not know the private key SK_c, they will not get the key K, so that the token is not available, then our system is still safe.
4) **Prevent the single point of failure caused by CSP itself:** Assume that the CSP is faulty, then the system will directly exit the system in the smart contract, which will not cause loss to our system, or the user can immediately execute the smart contract SC1. The kill() method is used to destroy the code, which also ensures that your rights are not affected.

VII. CONCLUSION AND FUTURE WORK

The motivation for establishing an identity management system based on Ethereum's smart contracts is to solve the problem of excessive third-party-centricity of existing identity management systems, and our new solution enables users and CSPs to supervise and manage the system through Ethereum Smart Contracts jointly. In the new scheme of this paper, the latest technology of blockchain used as a powerful tool in the user authentication step in the identity management system, which makes the blockchain technology help us to supervise each other and jointly promote the security of the network. In our EIDM protocol, the privacy of both users and CSPs also considered, and users are more flexible, and the interaction between users and CSPs becomes fairer. With the development of blockchain technology, Ethereum technology will continue to grow. Our solution is to let cloud users realize their identity management through Ethereum technology. The reason why the Ethereum blockchain is used to replace the previous IDMs system is because our system has the Ethereum blockchain record of making the results of our operations transparent,and through JWT and cryptography knowledge, we make our The operation process is relatively private compared to the previous one, which respects cloud users and appreciates cloud service, providers. On the other hand, users and cloud service providers can transmit data themselves through smart contracts, and the flexibility between the two is also improved. In our experiments, we can also see that the cost of smart contracts is not very high, it has individual practicability, and our new protocol also has less data transmission than the previous protocol and improved the earlier protocol of data transmission.

In a future work, our program is still facing some of these problems. First, because Ethereum mining takes time, our system needs to improve the speed of user login. Also, we should use more advance encryption methods to increase the speed of encryption and decryption of user information. Finally, since the protocol of the EIDM system is tested based on the CIDM protocol, our system can also be used on smartphones, and only open any browser application in the smartphone to input our test address [24].

You can see the interface of the system, but since the Ethereum wallet is temporarily not installed on the plug-in of the smartphone browser, there is still a lot of work to be done on the interactive function of the smartphone.

APPENDIXES

You can find those address in the Rinkeby Testnet of Ethereum.
CDC Ethereum Wallet Address:
0x1c1c265f4b4da8247e8df5c0ef07c757ee07b3bb
User's Ethereum Wallet Address:
0x0A04B00C07ADDC1d24D463074A5BF7897A0a9E5f
Csps contract address:
0x316a7a3C4760E5d345bC9046Bbe32a07075cB518
Users contract address:
0x06ba741090dc310f587a86841c31448375391002
Signature contract address:
0x48a1ceb52f6e19076ae1273d30bc4025ca527008

ACKNOWLEDGMENT

Thanks also go to the anonymous reviewer for their userful comments.

REFERENCES

[1] R. Shaikh and M. Sasikumar, "Identity management in cloud computing," *Int. J. Comput. Appl.*, vol. 63, no. 11, 2013.
[2] P. Angin, B. Bhargava, R. Ranchal, N. Singh, M. Linderman, L. B. Othmane, and L. Lilien, "An entity-centric approach for privacy and identity management in cloud computing," in *Proc. 29th IEEE Symp. Reliable Distrib. Syst.*, Oct./Nov. 2010, pp. 177–183.
[3] L. Yan, C. Rong, and G. Zhao, "Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography," in *Proc. IEEE Int. Conf. Cloud Comput.* Berlin, Germany: Springer, 2009, pp. 167–177.
[4] I. Khalil, A. Khreishah, and M. Azeem, "Consolidated identity management system for secure mobile cloud computing," *Comput. Netw.*, vol. 65, no. 2, pp. 99–110, Jun. 2014.
[5] O. Acobovitz, "Blockchain for identity management," Dept. Comput. Sci., Lynne William Frankel Center Comput. Sci., Ben-Gurion Univ., Eilat, Israel, Tech. Rep., 2016, vol. 1, p. 9.

[6] M. Jones, B. Campbell, and C. Mortimore, *JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants*, document RFC 7523, 2015.

[7] M. Suguna, R. Anusia, S. M. Shalinie, and S. Deepti, "Secure identity management in mobile cloud computing," in *Proc. Int. Conf. Nextgen Electron. Technol., Silicon Softw. (ICNETS2)*, Mar. 2017, pp. 42–45.

[8] S. Kirkman and R. Newman, "A cloud data movement policy architecture based on smart contracts and the ethereum blockchain," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Apr. 2018, pp. 371–377.

[9] D. Augot, H. Chabanne, T. Chenevier, W. George, and L. Lambert, "A user-centric system for verified identities on the bitcoin blockchain," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Cham, Switzerland: Springer, 2017, pp. 390–407.

[10] Y. Chi, G. Li, Y. Chen, and X. Fan, "Design and implementation of OpenStack cloud platform identity management scheme," in *Proc. Int. Conf. Comput., Inf. Telecommun. Syst. (CITS)*, Jul. 2018, pp. 1–5.

[11] J. K. Hanna, "Systems and methods for an incremental, reversible and decentralized biometric identity management system," U.S. Patent 10 078 758 B1, Sep. 18, 2018.

[12] A. Miller. (2018). *The Author Gives a Lot of New Ideas on the Identity Management System Field. Never Use Passwords Again With Ethereum and Metamask*. [Online]. Available: https://hackernoon.com/never-use-passwords-again-with-ethereum-and-metamask-b61c7e409f0d?gi=6def0cb49a0e#4ac4ifeb7

[13] A. W. Alrodhan and J. C. Mitchell, "Improving the security of CardSpace," *EURASIP J. Inf. Secur.*, vol. 2009, Mar. 2009, Art. no. 167216.

[14] PRIME. (2010). *Privacy Identity Management for European*. [Online]. Available: https://www.prime-project.eu/

[15] OPENID. (2010). *This is a Distributed Authentication Protocal*. [Online]. Available: http://openid.net/

[16] OAuth. (2007). *This is a Distributed Authentication Protocal That Help Cloud Users Manage Identities*. [Online]. Available: https://oauth.net/

[17] OAuth 2.0. (2011). *OAuth 2.0 is the Industry-Standard Protocol Forauthorization*. [Online]. Available: https://www.prime-project.eu/

[18] K. Khash. *Four Attacks on OAuth-How to Secure Your OAuth Implementation. The Author Gives Many Examples That are Close to Our Lives and Illustrates the Shortcomings of This Agreement*. Accessed: 2014. [Online]. Available: https://www.sans.org/reading-room/whitepapers/application/paper/33644

[19] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic CashSystem*. [Online]. Available: https://bitco.in/pdf/bitcoin.pdf

[20] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[21] *JWT*. Accessed: 2015. [Online]. Available: https://jwt.io/

[22] C. Dannen, *Introducing Ethereum and Solidity*. New York, NY, USA: Apress, 2017, pp. 89–110.

[23] (2019). *The Ethereum Identity Management Model*. [Online]. Available: https://github.com/flower-pp/ethereum-IDM

[24] (2019). *This is a Test Model Web. You Can See it on a PC or a Smart Phone*. [Online]. Available: http://47.103.11.164

[25] (2014). *Web3.js.It is a New Ethereum JavaScript API Which Connects to the Generic JSON RPC Spec*. [Online]. Available: https://github.com/ethereum/web3.js

[26] Rinkeby. *An Ethereum Test Net. You Can Check the All Transitions If You Can Kown Every Transaction's Address*. Accessed: 2019. [Online]. Available: https://rinkeby.etherscan.io

SHANGPING WANG received the B.S. degree in mathematics form the Xi'an University of Technology, Xi'an, China, in 1982, the M.S. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, in 1989, and the Ph.D. degree in cryptology from Xidian University, Xi'an. He is currently a Professor with the Xi'an University of Technology. His current research interests include cryptograph and information security.

RU PEI received the B.S. degree in mathematics form the Baoji University of Arts and Sciences, Baoji, China, in 2017. She is currently pursuing the M.S. degree with the Xi'an University of Technology, Xi'an, China. Her current research interests include information security and blockchain technology.

YALING ZHANG received the B.S. degree in computer science from Northwest University, Xi'an, China, in 1988, the M.S. degree in computer science, and the Ph.D. degree in mechanism electron engineering from the Xi'an University of Technology, Xi'an, in 2001 and 2008, respectively. She is currently a Professor with the Xi'an University of Technology. Her current research interests include cryptograph and network security.