Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5560621, 15 pages
https://doi.org/10.1155/2021/5560621

Research Article

Distributed Blockchain-Based Authentication and Authorization Protocol for Smart Grid

Yuxin Zhong,[1] Mi Zhou,[2] Jiangnan Li,[1] Jiahui Chen,[3] Yan Liu,[1] Yun Zhao,[2] and Muchuang Hu[4]

[1]*Shenzhen Power Supply Bureau Co. Ltd, Shenzhen 518001, China*
[2]*Electric Power Research Institute, CSG, Guangzhou 510663, China*
[3]*School of Computers, Guangdong University of Technology, Guangzhou, China 510006*
[4]*Department of Science and Technology, People's Bank of China, Guangzhou, China 510000*

Correspondence should be addressed to Jiahui Chen; csjhchen@gmail.com

Received 19 February 2021; Revised 10 March 2021; Accepted 31 March 2021; Published 23 April 2021

Academic Editor: SK Hafizul Islam

Authentication and authorization (A & A) mechanisms are critical to the security of Internet of Things (IoT) applications. Smart grid system processing and exchanging data without human intervention, known as smart grids, are well-known as IoT scenarios. Entities in such smart grid systems need to identify and validate one another and ensure the integrity of data exchange mechanisms. However, at present, most commonly used A & A protocols are centralized, resulting in security risks such as information leaks, illegal access, and identity theft. In this study, we propose a new distributed A & A protocol for smart grid networks based on blockchain technology to address with these risks. The proposed protocol integrates the decentralized authentication and immutable ledger characteristics of blockchain architectures suitable for power systems with a novel blockchain technique to realize both identity authentication and resource authorization for smart grid systems. We discuss the security of and threat models for prior A & A protocols and demonstrate how our protocol protects against these threats. We further demonstrate an approach to a real deployment of our A & A protocol using the FISCO consortium platform, applying algorithms from smart contract systems. Finally, we present the results of experimental simulations showing the efficacy and efficiency of our proposed protocol.

1. Introduction

Enabled by recent advances in monitoring, sensing, control, and communication technologies as well as the recently increasing popularity of renewable energy and distributed power resources, conventional power grids are presently developing along a path of becoming power information and communication networks, realizing a vision of self-healing and self-restoration capabilities and improved sustainability and efficiency, which have been commonly referred to as the smart grid (SG) [1]. In a smart grid network, power processing and data exchange do not require human intervention. Consequently, such highly automated systems require entities in smart grids to identify and verify one another to ensure the security of the data transactions. Otherwise, they become vulnerable as potential targets of malicious users. Because of the size and other characteristics of smart grids, efficient centralized authentication systems are nearly impossible to construct in practice. More precisely, in smart grid networks, traditional power grid dispatching management systems based on centralized architectures have been shown to encounter the following problems [2]: (1) the operation and supervision costs of trading centers have been shown to be relatively high and are directly transferred to the distributed power generation providers; (2) a natural trust problem exists between trade centers and connected traders, and it is difficult for trade centers to guarantee the fairness, transparency, and validity of information exchange; and (3) the centralized trading mode carries significant information security risks, threatening the security of transactions and the interests of traders. In contrast, blockchain technology uses asymmetric encryption, digital signature, negotiation, and other mechanisms to achieve peer-to-peer (P2P) transactions.

Thus, distributed authentication and authorization protocols present a potential solution to these problems. Authentication protocols define processes to validate user identities, whereas authorization protocols define processes determining what operations and functions users are permitted to perform once they are successfully authenticated. For example, in a smart grid, the identity of electricity users may be verified by providing a correct username/-password in a power information management system. Once a user has been authenticated, the power service provider can further authorize them to perform various functions if the system defines access policies describing user permissions. These issues are not unique to electrical power grid systems; further examples of authentication and authorization include many other methods, such as the OpenID Connect [3] layer, Security Assertion Markup Language (SAML) [4], Oauth 2.0 [5], and JSON Web Encryption (JWE) [6]. However, all of these technologies share a common limitation, in that they all require a trusted third party. Authentication functions require a trusted third party to check credentials or other information provided by users to determine whether they are in fact who they claim to be. To implement authorization mechanisms, a trusted third party is required in centralized systems to confirm user authorization and determine what resources may be accessed by users and what permissions they may be granted on the system.

In contrast to centralized models, blockchain technology has developed rapidly since the emergence of Bitcoin in 2008 [7]. Owing to its decentralization, traceability, and tamperproof immutability, blockchain technology has attracted considerable interest from both academic and industrial researchers for its diverse applications to the construction of distributed systems. For example, many applications based on blockchain technology have emerged in the fields of crowdsensing [8] and the Internet of Things [9]. The trustless nodes in blockchain networks can maintain a consistent distributed ledger recording tamperproof transactions. From the perspective of system design, a blockchain must contain a consensus protocol to address the issue of agreement among trustless nodes. Because the decentralized blockchain architecture is well-suited to original distributed network designs, it has also attracted considerable attention for its potential to fulfill a wide variety of functions (tasking [10] and key management [11]) inspiring promising blockchain projects such as XRP [12], ETH [13], and Cosmos [14]. In general, blockchains are decentralized, shared ledgers with tamperproof, nonforgeable, and traceable characteristics. They combine blocks of data into specific data structures in a chronological and chained manner, adopting cryptography to ensure security. A number of studies have sought to construct decentralized A & A protocols using blockchain techniques [15], including Blockstack [16], Nym credentials [17], bubbles of trust [18], mutual authentication [1], and universal A & A [19]. All of these approaches seem promising, as they can offer significant benefits, including (1) decentralized naming systems supplying unique identities on a blockchain and (2) decentralized data exchange marketplaces on which cryptocurrencies, digital tokens, or data credits may be traded. However, they have several limitations. First, they are difficult to integrate with existing centralized applications. In general, developers must rebuild existing applications to embed these blockchain-based A & A protocols. Second, most of the relevant works do not consider authorization using blockchain techniques. Finally, all of them were developed using public blockchains such as Ethereum [13], resulting in inefficient processes.

In summary, three major challenges remaining to be solved in blockchain-based A & A protocols are described below:

(1) Security solutions to address potential threats in a smart grid system still require further development. On this basis, we review the current security and threat models and aim to develop a protocol addressing the implications of these models

(2) Protocols improving the flexibility of A & A processes in smart grid-related applications still need to be developed, and protocols that can be relatively easily deployed in mature platforms may be required

(3) Methods for improving the efficiency of blockchain-based protocols are also needed. This may require the development of blockchain-based protocols able to process more transactions in a given time (commonly quantified in transactions per second). For example, Ethereum 1.0-based blockchain applications support only 25-30 transactions per second at most, while some may be limited to roughly 15, which are all considered to be slow, whereas the speed of E-transaction in the conventional banking system is over 5000 transactions per second

1.1. Our Contribution. This study addresses the above challenges by proposing a new A & A protocol based on a consortium blockchain.

To summarize, the main contributions of the study are threefold:

(1) First, we propose a blockchain-based decentralized authentication and authorization protocol. We note that few works have considered authorization using blockchain techniques. To address security and common threats in smart grid systems, we discuss the security and threat models applicable to blockchain systems and analyze our proposed protocol approaches to the prevention of various attacks or exploits described in these models. In addition, to enrich the present work, we also discuss methods to enhance the security of blockchain implementations

(2) Our protocol is the first A & A protocol constructed using a consortium blockchain technique, as far as we know. We highlight that our protocol is expected to be easy to deploy on mature platforms such as the Hyperledger [20] or FISCO [21] platforms and is thus more flexible in developing applications for smart grid systems

(3) Lastly, we demonstrate an approach to real deployment of our A & A protocol using the FISCO platform, by patching algorithms used to maintain and record smart contracts. We then give the results of experiments to show the efficiency of our proposed protocol

1.2. Organization. The remainder of this paper is organized as follows. In Section 2, we introduce the necessary background knowledge. In Section 3, we review the relevant literature on blockchain-based A & A protocols considering both authentication and authorization. In Section 4 and Section 5, we present the main constructions of our proposed method and a security analysis. We then provide a description of a real deployment and an experimental evaluation in Section 6. Finally, in Section 7, we present a brief conclusion and discuss possible directions for future work.

2. Background

2.1. Consortium Blockchain. Blockchain technology is usually divided into public, private, and consortium chains according to the access mechanism [22] used in different application scenarios or system designs. Nodes on public chains are able to freely enter or exit the blockchain network and read or transfer data on the chain, whereas the access and transaction operations of private chain nodes are controlled by internal permissions, while read operations can be selectively public to the outside world. The distributed storage architecture of blockchain technology remains available for private chains. Consortium chains differ from public and private chains, and use a variety of media. These models fall on a spectrum between public and private chains by pre-setting participating nodes and permission controls. Owing to their relatively limited numbers of nodes and organized structure, consortium chains are characterized by weak centralization, strong controllability, strong expansibility, and rapid transaction speed, among other beneficial properties, which are mainly suitable for the applications in systems constructed by specific organizations or companies, i.e., power grid companies typically use data on electricity. Power grid technicians can control data access and quantities of power transmitted by setting permissions. On such systems, data transactions do not require the consensus of the entire network of nodes and such blockchains have characteristics of weak centralization. Thus, consortium blockchains meet electrical power data storage requirements of controlled access, efficient storage, and trusted storage.

2.2. Smart Contract. Smart contracts refer to computer programs that a network of mutually distrusting nodes can execute without the assistance of any trusted authority [13]. Compared to traditional programming source code, smart contracts use blockchain-immutable, distributed storage and tend to have special features such as enforcement mechanisms. In the initial stages of constructing an electrical energy data storage system, the power grid technicians can write trigger implementations according to the actual demand line contract conditions and contract content, in the form of scripts deployed to partners in the electrical energy data storage system. Once the system is operational, the contents of smart contracts can be executed when trigger conditions are met, completing data upload, network access, and other processing functions. In smart grid system based on blockchain technology, power grid technicians are expected to be able to write trigger actuators according to the actual requirements of contract conditions and contract contents, in the form of scripts deployed to partners in the electrical energy data storage system. Data can be guaranteed through each of the above characteristics. Finally, smart contracts can be made to implement a smaller granularity of permission control.

2.3. Security Requirements. To ensure the sustainability and scalability of automated grid ecosystems, an A & A protocol must fulfill numerous security requirements [18]. Therefore, motivated by this work, we describe the main security objectives for use cases in smart grid systems and introduce criteria commonly used to evaluate the security of authentication and authorization schemes.

Integrity: integrity is a critical requirement for smart grids. As far as we are concerned, integrity assurance can be divided into two aspects, including (1) message (transaction/communication) integrity: during transmission or communication over the whole network, messages exchanged shall not be changed or modified; (2) data integrity: the consistency and reliability of data in smart grid networks are maintained throughout their life cycles. Therefore, such applications require that only authorized users are allowed to recompose the stored data.

Availability: availability means that the resources should only be accessed on demand by legitimate users. Therefore, a smart grid system must be capable of resisting denial-of-service (DoS) attacks to ensure authentication services.

Scalability: in the context of smart grids, scalability refers to the capability of ensuring that the size of a system has no effect on its performance. For instance, if the resources used by the user base increase dramatically, the time required for system service functions such as authentication or authorization must not be affected.

Undeniability: undeniability is the requirement that any user cannot deny that they performed a given action; for example, a user cannot refuse to accept that they sent some message.

Identification: identification is the primary requirement in most smart grid use cases. This requirement represents an absence of anonymity to ensure that any user can utilize the functions of the system. For instance, in a smart meter billing scenario, once a sensor sends a notification, the grid management system must know exactly which sensor was the source of the communication to accurately update the corresponding smart meter.

Mutual authentication: mutual authentication means that both parties need to put forward the certification requirements for each other. This requirement plays an important role in preventing systems from being attacked by spoofing entities.

Authorization: authorization specifies what a user is permitted to do. In most smart grids, authorization is usually considered as a method of establishing access to resources, such as power, computational power, or network bandwidth. Authorization protocols can also handle users' privileges on the system or network, including whether a particular user has access to the system at all.

2.4. Threat Models. Motivated by the related work [18], in this section, we present some anticipated possible threats, focusing specifically on blockchain networks used for smart grid applications. We assume a potential attacker to have full control over the network used; that is, they are able to arbitrarily sniff, discard, reorder, replay, inject, defer, and tamper with messages with negligible latency. More precisely, an attacker can take various kinds of threat or attack actions, such as sending false error messages to mislead a system decision or deny a system service. The following attacks can be performed:

DDoS attacks. Nodes can be subjected to distributed denial-of-service (DDoS) attacks, causing them to malfunction. For example, if a hostile threat actor sends a massive amount of small transaction requests to a node, the node may not be able to process normal data transactions. Another example is that using a node's memory through some vulnerability may cause a node to crash.

51% attacks. An attacker controlling 51% of the resources of an entire smart grid network can create new branches and replace the original branch of the chain. Using this mechanism, an attacker can add blocks with forged transaction data to their new forged chain and invalidate transaction data in the original fork. In this manner, an attacker could make illegal profits, which could lead to further economic loss.

Double-spending attack. Double-spending attacks result in incorrect transactions. By controlling 51% of the computational power of an entire blockchain network, an attacker can create a new branch and roll back transactions. A subsequent transaction then requires the same asset a second time, allowing an attacker to steal the assets from the first transaction.

Sybil attacks. In blockchain-based smart grid networks, a Sybil attack can be executed on two sides. First, Sybil attacks on the consensus mechanism refer to an attacker wasting computational resources of other nodes by controlling most nodes in the network and thus controlling mining and consensus operations in the blockchain network. In the POW consensus mechanism, the agreed node must immediately broadcasts a newly discovered block to the entire network. However, in a Sybil attack, an attacker might waste computing resources on other nodes by not immediately reporting to the entire network when they find a new valid block but instead continue to mine and attempt to mine further blocks before submitting the previously mined blocks. When another node produces a new block, an attacker immediately broadcasts its stored valid block to the entire network, rendering the mined blocks of the other nodes invalid.

Alternatively, Sybil attacks on the network refer to a malicious node weakening or destroying the redundant backup mechanism by simulating or controlling multiple nodes. Under this attack, multiple data backups may be stored on the Sybil nodes and thus potentially corrupted or compromised.

Smart contract vulnerability attack. A smart contract vulnerability on the blockchain is visible to all nodes that have access to the blockchain but cannot be fixed quickly as a result of the immutable characteristics of blockchain systems. Therefore, the security of smart contracts is critical. However, many smart contract security issues have been found, such as integer overflows, storage corruption, and logic vulnerabilities. These vulnerabilities could result in an attacker taking full control of tokens in a smart contract or acquiring a large number of tokens that do not belong to them, which would compromise the information security of both the contract owner and the user.

Eclipse attacks. An Eclipse attack involves an attacker controlling a sufficient number of addresses to monopolize all connections to a target blockchain node. Attackers can then use these victims to attack blockchain mining and consensus systems, including N-confirmed double payments, forged mining, and adversarial forks in the blockchain system. For example, a fake mining attack is often used in which an attacker employs multiple controlled IPs for the purpose of fake mining, that is, "mining" without providing any computational power, while the system incorrectly assumes that significant computational power is provided by the controlled IP addresses.

Spam attacks. If many transactions are sent to a blockchain network in a short period of time, the nodes in the network must then receive, validate, transmit, and store data; thus, blockchain networks can be flooded with overwhelming data packet traffic. This can cause the connection bandwidth to oversaturate and the system to reach its resource limits. In this type of attack, an attacker can send a large number of specific garbage transactions that serve no purpose other than populating the blockchain network and blockchain nodes. For example, an attacker could send large numbers of transactions with tiny amounts of digital currency that could not be used for actual payments, but which could cause temporary congestion or denial of service.

3. Related Work

In this section, we review related works on both authentication and authorization using blockchain techniques.

Fromknecht et al. [23] proposed the first distributed public key infrastructure (PKI) system based on the technology of a blockchain named CertCoin in 2014; the core idea in their work was to associate the form of public user identity with a public key certificate through a public ledger record, so as to realize decentralized construction of PKI, enabling any user to query the issuance process of a certificate. Their work demonstrated the capability to directly solve the traditional PKI system bottlenecks of the certificate transparency problem and the certificate authority (CA) single point of failure problem. However, because transaction information on a public blockchain is publicly visible to anyone, CertCoin's method of directly linking a public key to a user's real identity is not applicable to settings in which user identity or privacy needs to be protected, such as vehicle-based networks and the Internet of Things.

Based on this, Axon and Goldsmith [24] improved the CertCoin model and proposed a privacy-preserving technology with a blockchain-based PKI model, named PB-PKI (privacy-aware blockchain-based PKI). Their model was different from traditional PKI, which directly links users' real identities through their public keys but protects the online key through associated offline keys to avoid the need for users to supply their real identities. Simultaneously, the authors divided user privacy levels into global and adjacent privacy and provided different levels of privacy disclosure according to different application scenarios. For trusted adjacent nodes, a user's real identity is directly associated with a public key. Anonymized identity authentication is implemented for untrusted global node disassociation, reducing the risk of user information privacy leaks.

In contrast to the above blockchain-based distributed PKI system, Matsumoto and Reischuk [25] studied the problem of insufficient investment to resist attacks by a CA in a given PKI system. Based on the financial characteristics of blockchain platforms, a PKI framework called Instant Karma PKI (IKP) [25] was proposed. Based on the Ethereum platform, IKP encouraged CAs to issue certificates correctly through economic means, introducing detectors to detect and report invalid certificates, and imposing economic penalties on CAs that issued such certificates. Participating HTTPS domains issued domain certificate policies (DCPs) to specify standards required for TLS certificates on the domain. Certificates that violated the DCP requirements were defined as invalid certificates. At the same time, member CAs could sell a reaction policy (RP) to the domain, meaning that if an illegal certificate was reported, the CA that issued the illegal certificate was required to make a payment transaction based on a smart contract, similar to financial insurance, to compensate the user. In addition, it was necessary for the detector to report invalid certificates. If a certificate reported was indeed invalid, the detector received a reward related to the number of such certificates detected, which could effectively avoid the detector reporting all CA certificates to obtain a higher reward. In [18], Hammi et al. proposed a primitive decentralized system, called "bubbles of trust," guaranteeing reliable identification and authentication of devices and protecting the integrity and availability of data. To achieve this, their approach relied on the security advantages provided by blockchain technology and was used to create secure virtual areas (bubbles) where communications could be identified and trusted.

For distributed PKI blockchain architectures, Hari and Lakshman [26] realized Border Gateway Protocol (BGP) routing propagation and authentication of trusted nodes in DNS mapping to prevent malicious node attacks. Faisca and Rogado [27] solved the problem of identity management and certificate management in a personal cloud environment, improved the efficiency and security of authentication prior to interoperation between different clouds, and maintained the security and reliability of authentication consistent with that of blockchain technology. Kuo and Ohno-Machado [28] applied blockchain PKI to trust establishment in wireless sensor networks. Zhang et al. [29] addressed security authentication, data sources in intelligent contracts through front-end blockchains, and backend Software Guard Extension (SGX) trusted hardware implementation for data source authentication. Bassam [30] added authentication for attribute information by observing that the current x.509 certificate standard allowed certificates to be issued only for a user's identity but could record fine-grained identity certificate attribute information. In this approach, if a user's identity information was authenticated, the corresponding attribute of their identity was also considered trustworthy, realizing an exchange of trust between the user identity and a user attribute.

Some scholars have studied the authorization of cryptocurrency based on blockchain technology. Blockchain architectures have good anonymity, which can help users hide their real identity and effectively protect users' privacy information in public chains. However, in the fields of digital currency and private chain/alliance chains, it is necessary to ensure the authenticity of the node's identities in the blockchain network, because excessively strong anonymity may be expected to shelter illegal activities such as money laundering, drug trafficking, and malicious attacks within an organization. Based on this, Thomas and Pentland [31] explored proprietary trading chains and combined blockchain problems and identity authentication in the ChainAchor framework [32]. Based on zero-knowledge proof, the ChainAchor approach provides anonymous blocks in the chain of entities but can still verify the identity of the certification service. Real users can preserve a more valid identity and can selectively identify themselves in the process of trading. At the same time, ChainAchor provided prevention services for entities to initiate transactions, read transactions, and verify transactions on an authorized blockchain. Consensus nodes managed a shared licensed blockchain by searching through a (read-only) list of anonymous members' public keys.

4. Proposed Protocol

4.1. Decentralized Authentication. In this subsection, we present our proposed general authentication protocol to build validated connections in a secure and decentralized manner via blockchain technology. As shown in Figure 1, our authentication protocol is composed of two processes, including registration and login.

(1) Registration. In the registration process of our authentication protocol, we first initialize a user's cryptographic public key PK as their identity key. Then, we bind this identity key to a user-defined unique UserName and upload this binding on a blockchain, in which transaction can be verified later by other users. Finally, the user generates an identity transaction after the registration process. The list of information in the identity transaction of our designed registration process is shown in Table 1.

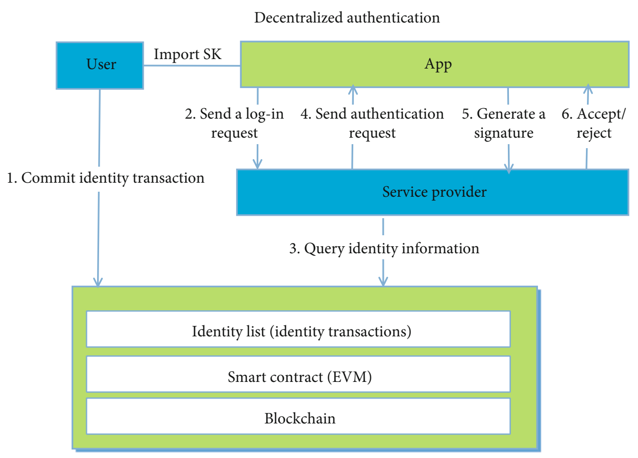
Decentralized authentication
User
Import SK
App
2. Send a log-in request
4. Send authentication request
5. Generate a signature
6. Accept/reject
1. Commit identity transaction
Service provider
3. Query identity information
Identity list (identity transactions)
Smart contract (EVM)
Blockchain

Figure 1: Decentralized authentication.

Table 1: Identity transaction.

Information	Description
TransactionType	Identity
User	An organization, an application, or a person
Identity	Identity card, passport, driver's license
PK	e.g., fcf4a1f566d1e0aa06436098c09d 35d9762bf240
UserName	Alice001
Timestamp	The time the transaction occurs, i.e., 177131cee76
Signature	e.g., 0xc3373d3bd1d4edc089001fd330920c303e 95c51b131c22bc91b2f9f9f56e0de9

TransactionType. The transaction type variable refers to the transaction in which the blockchain activity actually takes place, which occurs only between designated users who sign with the private key of each user. Here, the transaction type is stored in the block and cannot be altered or forged after the authentication process has been completed.

User. A user refers to any entity that uses the A & A service in a smart grid system.

Identity. Identity defines a unique representation of a user in a smart grid system. This functions like a personal identity card, passport, driver's license, etc.

PK. The public key PK in a blockchain usually acts as an address string and plays the role of an account. In our design, PK plays the role of user identity. In addition, the PK is uploaded to the blockchain, which is open to everyone.

UserName. PK is usually too long or too random to be human-readable. Therefore, for a better user experience, in our protocol, we allow users to select a unique human-readable UserName and binds this name to their PK. Other users can use this unique UserName to look up the user's PK and other information in the above list shown in Table 1. Currently, many blockchain technologies can implement this type of binding. For example, in the Blockstack project, a smart contract using an EOS can bind PK to a human-readable UserName. In addition, they establish a DNS-like system to provide naming services, and users can obtain other users' PK and other relevant information through their UserNames. We also adopt these methods in our decentralized protocol, in which any peer with a full copy of the blockchain data can provide a naming service to users.

Timestamp. In our protocol, timestamps are used to authenticate each transaction record, which can show the authenticity of the transaction record, similar to the certification of the transaction contracts. As part of the block metadata, the timestamp provides a natural time record for the transaction sequence.

Signature. It is important to ensure that the contents of a transaction are not modified by signing. Because transactions are public, anyone can access all of them, even if they have not yet been added to the blockchain.

(2) LogIn. After registration, a user logs in to the smart grid system to use a service by running the "LogIn" process. The process flow is shown in Figure 1 and is described as follows:

(1) To log in, the user first commits the identity information and imports their secret key into the service application. In this case, the secret key can be imported by typing it manually or scanning by a QR code, etc.

(2) A user who needs to log in then sends a login request to the service provider of the smart grid network. Note that the login request is signed with the secret key SK of the user logging in

(3) The service provider parses the login request, extracts the hash, queries the blockchain, obtains identity information as PK/UserName from an identity list (identity transactions), and runs the associated smart contract as described below

(4) Upon finishing the above process, an authentication request is sent back by the service provider. The authentication request contains a timestamp (to prevent a replay attack), the user's UserName, and their PK, along with a signature

(5) The user generates a signature with five parameters, including timestamp, UserName, and PK, as well as the service provider's UserName and PK. The signature is treated as the user authentication credential and is placed in the header of the transaction

(6) The service provider verifies the received information, and if it is valid, the authentication succeeds; otherwise, the authentication fails, and the service provider denies the user login

After the LogIn process, the service provider obtains further personal information from the user to build their profile, which will then be used in the authorization mechanism.

(3) Smart contract design. Here, we describe our proposed smart contract algorithms to further show how to realize our protocol. First, we need two key

```
Input(accountId)
Output Record or None
1:  Table table = openTable("Identity");
2:  Entries entries = table.select(accountId);
3:  if 0 == uint256(entries.size()) then
4:      return None
5:  else
6:      return Record
7:  end if
```

Algorithm 1: QueryData.

```
Input(accountId, publicKey, userInfo, sigData)
Output True or False
1:  ret = Query(accountId)
2:  if ret != None then
3:      return False
4:  else
5:      Table table = openTable();
6:      Entry entry = table.newEntry();
7:      entry.set("accountId", accountId);
8:      entry.set("publicKey", publicKey);
9:      entry.set("userInfo", userInfo);
10:     entry.set("sigData", sigData);
11:     table.insert(accountId, entry);
12:     return True
13: end if
```

Algorithm 2: SendTransactions.

functions, namely "QueryData" and "SendTransaction." The smart contract algorithms are as follows.

As shown in Algorithm 1, we use the "QueryData" function to query the identity or resource lists in the blockchain. This smart contract utilized the CRUD interface to simplify programming, similar to executing an operation on a relational database. A programmer who wishes to write a smart contract is able to obtain the identity or resource transactions as a list stored in a table and then query the target record by the key-value "accountId." In the decentralized application, the service provider calls the "QueryData" function to query a user's identity information when it receives a user login request.

As shown in Algorithm 2, we use the "SendTransactions" function to send a transaction to the blockchain. Specifically, the user can send his/her identity information to the blockchain using this function in the registration process of the authentication protocol. The blockchain system checks the validity of registration after it receives the transaction. If the identity information is valid, the data is then written to the blockchain.

Remark 1. Note that we do not give the entire transaction of the input due to space limitations. According to the transaction, we may observe the deployment process. In our proposed approach, when a smart contract is deployed, the code, written in Solidity, is compiled into a binary string. Later, the binary string is encoded with the recursive length prefix (RLP) encoding and constructed as the input of the transaction. The address in the "from" field is the public key of the user who deployed the smart contract. They then sign the transaction with their private key. Finally, the transaction is sent to the blockchain.

```
{
blockHash: 0x07632ed9b2dddf8ceb7cdef0c100a2cd5dc
0d9a1f483c1f9b7f7db66d7a783fd,
blockNumber: 0x1bc72
gas: 0x419ce0
from: 0x35ee5b080153b91c77db3d5c07c17daecf018d76
transactionIndex: 0x0
to: 0x0000000000000000000000000000000000000000
nonce: 0x17e46d634352dcbb206b0d0d3dcfe8542d01d62
0692ba94609af70a700cd9e7
value: 0x0
hash: 0x3f4de7ca0aa62adcbfcbbe46b40f94be34f9a3cd
56849f5f2dad811f6953d127
gasPrice: 0x51f4d5c00
input:0x608060405234801561001057600080fd5b5061002
861002d640100000000026401000000009004565b
}
```

Algorithm 3: The smart contract transaction.

Thereafter, to achieve decentralized authentication, we must solve two types of tasks. One involves calling the abovementioned smart contract to access the blockchain system, while the other involves handling interactions between users and service providers. The decentralized authentication mechanism consists of "Registration" and "Login" algorithms.

As shown in Algorithm 4, we randomly generated the identities and public keys of users to simulate the registration process. If an account id already exists, the blockchain system is designed to reject the registration and increment the number of failures.

As shown in Algorithm 5, we randomly selected the identities of some test users to simulate the login process. The account id we chose could be nonexistent, and by design, the service provider should reject a user login if the verification fails. When the service provider receives a login request, they first check the validity of the account data on the blockchain. If it is valid, the service provider gives a challenge for the user to sign. The user generates a token signed by their private key. Finally, the service provider verifies this token and decides whether the user has permission to log in.

4.2. Decentralized Authorization. As is shown in Figure 2, our proposed authentication protocol is composed of two parts, including resource registration and resource granting.

(1) Resource registration. In this process, a user registers basic information on the blockchain of their resource. Suppose that a user on the smart grid requests a

```
Input(AccountId, UserInfo)
Output True or False
1:  Get cryptographic key pair cryptoKeyPair = createKeyPair();
2:  String PublicKey = cryptoKeyPair.getHexPublicKey();
3:  String data = AccountId + UserInfo + PublicKey;
4:  Get cryptographic suite cSuite = new CryptoSuite (CryptoType.ECDSATYPE);
5:  String hashData = cSuite.hash (data);
6:  String sigdata = cSuite.sign (hashData, cryptoKeyPair);
7:  rec = register (UserInfo, PublicKey, UserInfo, sigdata);
8:  response = getRegisterEventEvents (rec);
9:  if response. get (0) ==0 then
10:     regFail++£» True
11:     return True
12: else
13:     return False
14: end if
```

Algorithm 4: Registration.

resource of the resource server, and the user needs to submit a resource transaction to the blockchain through the blockchain client. This resource transaction should include at least the following information.

(2) Resource granting. In this process, the user implementation requests the resource of the resource owner and obtains authorization. Suppose that a user requests access to a resource owned by a resource owner, and the resource owner decides whether to grant access issuing a public-key cryptographic signature. This process is illustrated in Figure 2 and described as follows:

(i) In the beginning, the resource owner first records their resource on the blockchain using the smart contract algorithm "RegisterResource" described below. Then, the requesting user sends an authorization request to the resource owner with parameters including the user's UserName and its PK, as well as the user's information, resource address, and timestamp. Note that the request is signed using the user's secret key, SK

(ii) Upon receiving the request, the resource owner then verifies the validity of the request by checking the validity of the signature and decides whether the user who made the request may access the resources they own. This process is performed using the smart contract algorithm "GrantResource," which is described below. If the verification passes, indicating that the resource owner decided to grant authorization, they release an access token based on the user's previous request. The access token is in fact a new signature with a timestamp, UserName, user's PK, resource address, timestamp, and so on

(iii) Upon receiving the access token, the requesting user further sends the actual resource access request to the resource server. Because the access token is available, the resource server can establish that the resource owner has already granted the requesting user's access request and proceeds to the step below

(iv) Upon receiving the parameters from the previous request, the resource server parses and verifies the validity of the request's signature. If it is verified successfully, the resource server queries the owner's information by checking the resource list on the blockchain and then checks the validity of the access token. If this also passes, the resource server sends the requested protected resource to the user

(3) Smart contract design. Similarly, we designed smart contract algorithms to achieve decentralized authorization. The application also needs to call the smart contract to access the blockchain system, and the communication among the resource owner, requester, and resource server is ignored. The proposed decentralized authorization protocol consists of "RegisterResource" and "GrantResource" algorithms

Remark 2. Note that the resource management in our design is also a decentralized storage network using the InterPlanetary File System (IPFS) technique [33] or other decentralized cloud storage service providers.

As shown in Algorithm 7, every resource owner can use the "RegisterResource" algorithm to record their resources on the blockchain.

As is shown in Algorithm 8, there are three key processes in the "grantResource" procedure. First, the resource owner

```
Input(accountId)
OutputTrue or False
1:  $contractAddress = loadAuthenticationAddr()$;
2:  $auth = Authentication.load\ (contractAddress)$;
3:  $result = auth.select(accountId)$;
4:  if $result.getValue1() == 0$ then
5:     Verify the validity of the account data on the blockchain
6:     String $tmpPublicKey = result.\ getValue2()$;
7:     String $UserInfo = result.\ getValue3()$;
8:     String $tmpsigdata = result.\ getValue4()$;
9:     Get cryptographic suite $cSuite = new\ CryptoSuite\ (CryptoType.ECDSATYPE)$;
10:    String $tmpdata = accountId + UserInfo + tmpPublicKey$;
11:    String $hashData = cSuite.hash(tmpdata)$;
12:    Boolean $nVer = cSuite.verify(hashData, tmpsigdata)$;
13:    if $nVer == True$ then
14:       Generate a token to be signed by the user
15:       Long $timeStamp = System.currentTimeMillis()$;
16:       $Stringdata = accountId + timeStamp + getHexPublicKey()$;
17:       $sigdata = SignWithSecp256k1(data)$;
18:       Verify the validity of token data
19:       if $expireTime < (5 * 60 * 1000)$ then
20:          $data = cSuite.hash(data)$;
21:          $nVer = cSuite.verify(data, sigdata)$;
22:          if $nVer == True$ then
23:             return $True$
24:          end if
25:       end if
26:    end if
27: end if
28: return $False$
```

Algorithm 5: Login.

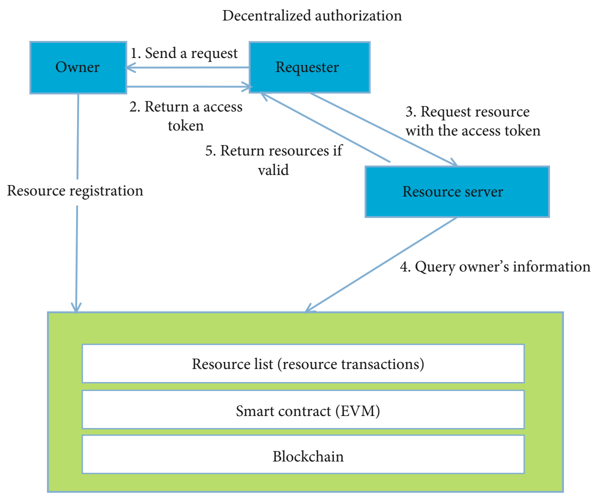
Decentralized authorization
Owner
1. Send a request
Requester
2. Return a access token
3. Request resource with the access token
5. Return resources if valid
Resource server
Resource registration
4. Query owner's information
Resource list (resource transactions)
Smart contract (EVM)
Blockchain

Figure 2: Decentralized authorization.

generates an access token with their signature when they receive the request from the user requesting the resource. The resource requester can use this access token to request the owner's resource from the resource server. Then, the resource server queries the owner's information on the blockchain as soon as they receive the token. Finally, the resource server verifies both the validity of the access token and the resource transaction to decide whether to grant the resource.

5. Security Analysis and Discussion

5.1. Security Analysis. We describe below the reasons that our protocol satisfies the security requirements mentioned in Section 2.3 and describe how it is expected to avoid vulnerabilities to the attacks mentioned in Section 2.4.

Mutual authentication and messages integrity: every entity in the smart grid system uses a "PK bind to UserName" method of secure authentication. This method is delivered to a valid entity only during initialization, and all message transactions are signed using the ECDSA algorithm (the secret key associated with the PK). Thus, such cryptographic signatures ensure mutual authentication of message integrity for sending and receiving devices, as well as other devices on the network. In addition, as stated in Section 2, blockchain technology should be considered reliable.

Identification: as is mentioned above, each entity in the smart grid system has an identity key. The trustworthiness of the identity system is ensured through identity keys on the blockchain. More precisely, each message object has its identity bound to a public key associated with it. Therefore, the system can easily recognize each object.

Undeniability: in our proposed protocol, each message is signed by a secret key, which is only known to its owning entity, which is also the only entity who can use it. Therefore, it cannot be denied that a given entity signed some message,

```
{
User’s profile,
Resource address: “ipfs://QmUgmeDNfbP61hynZAQjvy81sGo.
QmLQRzEaG51knUS9rjk”
UserPK:148947262ec5e21739fe3a931c29e8b84ee34a0f
b60b7b13d6a79f1ae5174
UserName:Bob001
Signature:0340a609475afa1f9a784cad0db5d5ba7dbaab
2147a5d7b9bbde4d1334a0e40a5
}
```

Algorithm 6:

```
Output True or False
1:  String data = UserResource + AccountId + PublicKey;
2:  Get ECDSA signature result sigdata = SignWithSecp256k1(data);
3:  rect = register(AccountId, PublicKey,UserResource, sigdata);
4:  response = auth.getRegisterEventEvents(receipt);
5:  if response.get(0) ==0 then
6:      return True
7:  else
8:      return False
9:  end if
```

Algorithm 7: RegisterResource.

and the undeniability criterion is satisfied by this cryptographic signature scheme.

Scalability: our system is constructed based on a consortium blockchain, which in fact is a peer-to-peer network organized by different nodes in a consortium. As pointed out in [34], a peer-to-peer network can efficiently solve the scalability problem for large sales.

DoS/DDoS prevention: the decentralized architecture of blockchain technology makes it highly resistant to DoS/DDoS attacks. Moreover, in some blockchains, such as Ethereum, transaction costs are high, so attackers are also discouraged from sending large amounts of money or cryptocurrency to pay for transactions. In addition, transaction packet size is also related to transaction prices.

51 % attack prevention: to defend against 51% of attacks, it is recommended to maximize computing power or profit resources. For example, with the POW consensus mechanism, the greater the computing power of the entire network, the higher the cost for an attacker to carry out a 51% attack. In our protocol, we use the FISCO consortium blockchain, which embeds a practical Byzantine fault tolerance (PBFT) consensus mechanism. As shown in [35], it is robust against double-spending attacks.

Sybil attack prevention: Sybil attacks on the consensus side are often concerned with blockchain systems based on PoW. Because our protocol uses the FISCO consortium blockchain, the potential consensus algorithm can thus prevent this type of attack directly.

Then, for Sybil attacks on the network side, in the designed protocol, each entity can have only one public key as its identity, and accordingly, it can have only one secret key associated with the public key. We can see that each message in our protocol is signed by the secret key to ensure integrity. Moreover, the public key that binds to a UserName can be verified by the system, so that any attacker cannot use a forged public key, thus directly preventing the abovementioned Sybil attacks.

Spam Prevention: in our protocol, all messages are treated as transactions, and we arrange each transaction by a timestamp requiring a consensus phase to take effect. Therefore, an attacker cannot replay spam messages because the consensus mechanism will reject them. Moreover, we recommend reading in more detail in Ref. [34] regarding a method by which a blockchain can protect spam attacks.

Eclipse attack prevention: to protect against Eclipse attacks, in our protocol, nodes store trusted IP addresses and deploy mechanisms to check for misbehaving nodes in the network. IP addresses that misbehave over the network may be disabled. In addition, nodes check incoming and outgoing connections to reduce the impact of Eclipse attacks. Therefore, we have a good solution to prevent Eclipse attacks.

Smart contract vulnerability attack prevention: to defend against smart contract attacks, highly controlled or simplified virtual machines need to be customized to run smart contracts. Certain functions, such as access to system resources, direct access to memory, and interaction with file systems, should be disabled in custom virtual machines. For example, some projects build a new virtual machine instead of using an existing machine, such as Java, to implement advanced security for virtual machines. Operational codes (opcodes) associated with access to system resources, direct access to

```
Input(accountId, PublicKey)
Output Resource or None
1:  Generate an access token with the owner's signature.
2:  Long timeStamp = System.currentTimeMillis();
3:  String data = accountId + timeStamp + PublicKey;
4:  Get ECDSA signature result sigdata = SignWithSecp256k1(data);
5:  The user sends the signed token to the server.
6:  Resource server queries the owner's information on the blockchain.
7:  longexpireTime = System.currentTimeMillis() - timeStamp;
8:  expireTime < (5 * 60 * 1000) then
9:     Query the owner's information on the blockchain.
10:      Authorizationauth = Authorization.load(contractAddress);
11:      result = auth.select(accountId);
12:      if result.getValue1() == 0 then
13:         String tmpPublicKey = result.getValue2();
14:         String resourceData = result.getValue3();
15:         String tmpsigdata = result.getValue4();
16:         Verify the validity of the access token
17:         if tmpPublicKey.equals(PublicKey) then
18:            Get cryptographic suite cryptoSuite = new CryptoSuite(CryptoType.ECDSATYPE);
19:            hashData = cryptoSuite.hash(dataToBeSigned);
20:            Boolean nVerification = cryptoSuite.verify (hashData, sigdata);
21:            String tmpdataToBeVrified = accountId + UserInfo + tmpPublicKey;
22:            String hashData = cryptoSuite.hash(tmpdataToBeVrified);
23:            Boolean nVerification = cryptoSuite.verify(hashData);
24:            if nVerification == True then
25:               Verify the validity of the resource data
26:               String tmpdataToBeVrified = resourceData + accountId + tmpPublicKey;
27:               hashData = cryptoSuite.hash(tmpdataToBeVrified);
28:               nVerification = cryptoSuite.verify(hashData);
29:               if nVerification == True then
30:                  return Resource
31:               end if
32:            end if
33:         end if
34:      end if
35:   end if
36:   return None
```

Algorithm 8: GrantResource.

storage, and interaction with the file system are not stored in the virtual machine.

Double-spending attack prevention: the most effective way to prevent double-spending is to wait for multiple confirmations before offering goods or services to a recipient. In particular, the probability of success of a double-spending attack decreases as the number of confirmations received increases. The longer the backcross transaction in the blockchain, the more blocks must be captured before a malicious chain can be accepted in the network. This limits the possibility that an attacker can modify the history of transactions in the chain. However, this method is not suitable for rapid payment systems. In our proposed protocol, to ensure that the rapid payment system is protected from the double-spending attack, we also use techniques such as a listening period, inserting observers, and forwarding double-spending attempts to detect double-spending attacks. In the listening time technique, the vendor monitors all received transactions during a listening period and delivers the product only after verifying that there were no double-spending attempts. When inserting observers, the vendor inserts a set of observers that will relay all transactions they receive from the network directly to the vendor. In this way, the vendor can observe more transactions in the network while listening. This increases the chance of detecting double-spending attempts. The forward double-spending attempt technique requires each blockchain peer to forward all transactions that attempt to double-spend rather than discard them so that the vendor can be notified in the event of a double-spending attempt. This increases the chances of detecting double-spending attacks. However, all existing solutions only make double-cost attacks more difficult but do not eliminate the possibility.

5.2. Discussion.

In the previous section, we presented a security analysis of our proposed protocol. Most of our security prevention is addressed by the potential security of the blockchain architecture itself; therefore, in this subsection, we discuss methods to enhance the security of the blockchain.

Smart contract security design: to ensure the security of smart contracts, highly controlled or simplified virtual machines need to be customized to run such programs. Certain functions, such as access to system resources, direct access to memory, and interaction with the file system, should be disabled in these custom virtual machines. For example, some projects, such as Java, build a new virtual machine instead of using an existing machine, to implement advanced security for virtual machines. Operational codes (opcodes) associated with access to system resources, direct access to storage, and interaction with the file system are not stored in the virtual machine.

Privacy prevention: some privacy prevention techniques can be applied in the blockchain systems to protect privacy, as follows: (1) zero-knowledge proof: zero-knowledge proof is evidence that proves the truth of a statement without revealing any additional information beyond the content of the intended proof. In a blockchain system, we can use zero-knowledge proof to prove a statement such as "this asset transaction is valid" without revealing any important information about the transaction itself; (2) ring signature: a message signed using a ring signature is signed by someone in a particular group of people. The security of ring signatures is that an attempt to determine which member of the group owned a key used to produce a given signature is computationally infeasible. In a blockchain system, we can use ring signatures to prove a statement such as "this asset transaction is valid" without revealing the actual sender; (3) asymmetric encryption: other asymmetric encryption techniques, such as public key encryption, functional encryption, and homomorphic encryption, can also be used in blockchain systems to protect user privacy.

Illegal content prevention: several solutions have been proposed to address the threat of illegal content in blockchain systems, including (1) Merkle Hash Trees: we can use a Merkle Hash Tree to render each block, with each node in the tree representing a transaction hash. Then, we can set up a committee or voting mechanism to remove illegal content while retaining the transaction hash. This technology allows our proposed method to remove illegal content and keep the blockchain irreversible; (2) artificial intelligence technology: miners can use artificial intelligence technology to detect illegal content, such as illegal pornography, and can then discard transactions that contain illegal content, rather than packaging them into new blocks.

Defense against criminal activities: some solutions have also been proposed to reduce criminal activity using blockchain systems. For example, Know Your Customer (KYC) policy is the process by which an enterprise verifies the identity of its customers and assesses the potential risk of illicit intent in business relationships. By strengthening KYC policies, criminal activity in the blockchain systems can be reduced. In addition, artificial intelligence technology can be used to identify criminal activity in blockchain systems.

Memory corruption vulnerability prevention: memory corruption vulnerability is a common problem in software engineering. There are many methods to reduce this risk, such as source code auditing, fuzz testing, and penetration testing. Both can be used in blockchain systems.

Table 2: Registration in authentication.

Total time (MS)	Generate signature (MS)	Send transaction (MS)
445	15	430

Blockchain as the target of the threat: blockchains require replication and synchronization of information over a network. Today, there are many different categories of consensus algorithms. Each of them has specific characteristics; thus, it is necessary to suggest which consensus algorithm is more appropriate in the context of the security of different types of blockchain networks. Different organizations adopting the same type of distributed ledger technology solution may face the need for crossledger transactions. To function properly within the network, a distributed ledger peer needs to know its current ledger state (this issue has been resolved in the local ledger). However, when crossledger transactions are required, we need to trust that each ledger will work properly, or we need some mechanism to crossshare the ledger's state so that other nodes can verify that the transactions are correct. In addition, the KYC principle should not only provide means of identity privacy protection but also provide identity concealment, anonymity, obfuscation, and other mechanisms.

Blockchain as a source of threat: blockchain is a large-scale distributed coordinated system that runs the risk of becoming a source of DDoS attacks due to defects, malicious behavior, and/or cascading events. Node coordination logic locally present in blockchain technology makes injection easier and malicious actions from intruders or "logic bombs" more difficult to detect. In addition, because of its highly dispersed nature, it may be difficult to prevent such misconduct by blockchains.

6. Experiments and Performance

Our implementation consisted of three parts. One was the construction of a blockchain system on which we could run the proposed decentralized authentication and authorization. The other was to program smart contract algorithms based on blockchain. The smart contracts should at least implement the basic functions consisting of sending transactions and retrieving data on the blockchain. Finally, we should implement other functions that do not need to interact with the blockchain, such as the interactions between users, resource servers, and requesters. Fortunately, the blockchain we choose has provided the Java SDK, offering APIs for node status, blockchain system configuration modification, and nodes to send transactions. The Java SDK achieves this by calling the basic functions implemented by smart contracts. Therefore, we first used a smart contract to implement the functions referring to the blockchain and then wrote a Java application to complete the entire experiment we as required.

Table 3: Login in authentication.

Total time (MS)	Query account data (MS)	Verify account data (MS)	Generate token (MS)	Verify token (MS)
31	5	12	11	3

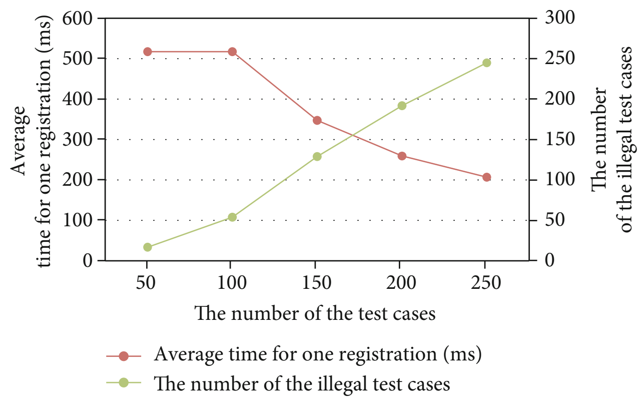

Figure 3: Registration test on the authentication process.

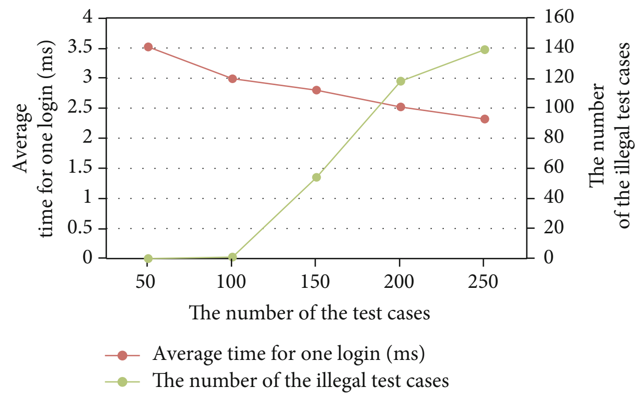

Figure 4: The login test on the authentication process.

6.1. Experimental Deployment. We deployed our blockchain system based on an open-source blockchain platform called FISCO BCOS. The code for our deployment is available in https://github.com/humuchuang/AAGrid. This platform includes are many features that can be configured based on the demand in the FISCO BCOS blockchain. Herein, we describe those configured in our blockchain. Our proposed blockchain is a secure, reliable, and consortium blockchain, and in single-chain installation, the performance has reached more than 10,000 TPS.

In our deployment, we chose cryptography algorithms described in [36] and distributed storage IPFS technique using the PBFT consensus algorithm [35], which is suitable for smart grid applications where different entities can function together as a consortium and they provide trust based on the blockchain consensus mechanism. Because it was available to deploy several different nodes on the same server for a test chain, we used a Linux server to deploy four nodes. The CPU of this server was a dual-core with an Intel (R) Xeon(R) Platinum 8269CY CPU @ 2.50 GHz on each core. The memory was 8 GB. We accessed the data on our blockchain using the CRUD interface (the Create, Retrieve, Update, Delete interface supplied by FISCO BCOS 2.0). For the smart contract of the blockchain, we chose the Java language. In addition, the blockchain system provides many cryptographic algorithms and protocols, such as the symmetric encryption algorithms AES and SM4, asymmetric encryption algorithms ECDSA and SM2, and asymmetric encryption elliptic curves secp256k1 and sm2p256v1 [37]. In our decentralized A & A application based on the blockchain system, we used the keccak256 hash algorithm [38] and the secp256k1 elliptic curve.

Table 4: Register resource in authorization.

Size of the resources (MS)	Total time (MS)	Generate signature (MS)	Send transaction (MS)
50B	500	14	486
100 B	532	15	517
1 kB	514	13	501
5 kB	537	17	520
10 kB	531	13	518

6.2. Experimental Performance. In our decentralized application, we focused on operations referring to access to the blockchain system and ignored other operations, such as communication between a user and the service provider. We ran 100 test cases for every application to obtain average results.

Decentralized authentication: as is shown in Tables 2 and 3, both the registration and login in the decentralized authentication were efficient.

Furthermore, in order to see the impact of illegal access, we wrote a simulation program to show the average time for sending one transaction to the blockchain system for different numbers of test cases. The number of test cases refers to the number of users attempting to register. As is shown in Figure 3, the average time to complete one transaction decreased when the amount of the invalid test cases increased.

Comparatively, we present simulation results for querying transactions on the blockchain system. Figure 4 shows the average time to query transactions one time on the blockchain system at different amounts of test cases. The number of test cases here refers to the number of reading information on the blockchain for the service provider to check the validity of the user. In general, the time required to read the data on the blockchain was less than that required to write transactions, and thus, decentralized authentication can be considered efficient.

Decentralized authorization: as mentioned above, in this study, we consider three types of resources, including computing power, data, and bandwidth. These resources have

Table 5: Grant resources in authorization.

Size of the resources (MS)	Total time (MS)	Generate token (MS)	Query resource (MS)	Verify the validity (MS)
50 B	29	11	5	13
100 B	26	11	4	11
1 kB	26	11	3	12
5 kB	30	11	5	14
10 kB	70	11	46	13

different sizes when recorded on the blockchain. Thus, a practical decentralized authorization should always act efficiently when the size of the resource data increases in a proper range. As shown in Table 4, we tested different cases in which a user registered their resources on the blockchain. The size of the resources ranged from 50 bytes to 10 kilobytes. We can observe that the time required to register resources on the blockchain changed slightly when the size of the resources increased. Similarly, Table 5 presents the results for granting resources. Generally, our decentralized authorization was efficient and the results demonstrate that it is suitable to be used in a practical smart grid for different types of resources.

As shown in Tables 4 and 5, both the registration and login in the decentralized authentication scheme are efficient.

7. Conclusion and Future Work

In this study, we have proposed a distributed authentication and authorization protocol for smart grid networks based on blockchain technology. Our proposed protocol applies a novel blockchain technique to realize terminal identity authentication in a smart grid system as well as communication authorization functions of the resources. In addition, it can reduce the security risks associated with centralization. We show in detail our proposed method to construct a distributed protocol and analyze its security vulnerability prevention capabilities. Finally, the deployment and experiments conducted demonstrate the efficiency of the protocol.

In future work, we plan to realize other functions of a smart grid system using blockchain techniques, such as data statistics and resource counting; additionally, we plan to extend our decentralized authorization function to a role-based access control function, and finally, we plan to develop a data privacy protection protocol using secure multiparty computing technologies, homomorphic encryption technologies, and agency encryption technologies, among other associated approaches.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] W. Wang, H. Huang, L. Zhang, and C. Su, "Secure and efficient mutual authentication protocol for smart grid under blockchain," *Peer-to-Peer Networking and Applications*, 2020.

[2] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber–physical system security for the electric power grid," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 210–224, 2012.

[3] M. Schwartz and M. Machulak, "OpenID connect," in *Securing the Perimeter*, pp. 151–203, Springer, 2018.

[4] H. Lockhart and B. Campbell, "Security assertion markup language (SAML) v2. 0 technical overview," *OASIS Committee Draft*, vol. 2, pp. 94–106, 2008.

[5] M. Jones and D. Hardt, "The OAuth 2.0 authorization framework: bearer token usage," Technical report, RFC 6750, 2012.

[6] M. Jones and J. Hildebrand, "JSON Web Encryption (JWE)," *Internet Requests for Comments, RFC*, vol. 7516, 2015.

[7] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, Manubot, 2019.

[8] K. Zhao, S. Tang, B. Zhao, and Y. Wu, "Dynamic and privacy-preserving reputation management for blockchain-based mobile crowdsensing," *IEEE Access*, vol. 7, pp. 74694–74710, 2019.

[9] E. K. Wang, Z. Liang, C.-M. Chen, S. Kumari, and M. K. Khan, "PoRX: a reputation incentive scheme for blockchain consensus of IIoT," *Future Generation Computer Systems*, vol. 102, pp. 140–151, 2020.

[10] Y. Wu, S. Tang, B. Zhao, and Z. Peng, "BPTM: blockchain-based privacy-preserving task matching in crowdsourcing," *IEEE Access*, vol. 7, pp. 45605–45617, 2019.

[11] C.-M. Chen, X. Deng, W. Gan, J. Chen, and S. K. H. Islam, "A secure blockchain-based group key agreement protocol for IoT," *The Journal of Supercomputing*, 2021.

[12] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: overview and outlook," in *International Conference on Trust and Trustworthy Computing*, pp. 163–180, Heraklion, Crete, Greece, 2015.

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

[14] J. Kwon and E. Buchman, "A network of distributed ledgers," *Cosmos, Dated*, pp. 1–41, 2018.

[15] P. Dunphy and F. A. P. Petitcolas, "A first look at identity management schemes on the blockchain," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 20–29, 2018.

[16] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: a global naming and storage system secured by blockchains," in *2016 USENIX Annual Technical Conference*, pp. 181–194, Denver, CO, 2016.

[17] H. Halpin, "Nym credentials: privacy-preserving decentralized identity with blockchains," in *2020 Crypto Valley Conference*

on Blockchain Technology (CVCBT), pp. 56–67, Rotkreuz, Switzerland, 2020.

[18] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, "Bubbles of trust: a decentralized blockchain-based authentication system for IoT," *Computers & Security*, vol. 78, pp. 126–142, 2018.

[19] T. Yang, W. Li, J. Yin, and Y. Deng, "A universal decentralized authentication and authorization protocol based on blockchain," in *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (Cyber C)*, pp. 7–14, Chongqing, China, 2020.

[20] V. Dhillon, D. Metcalf, and M. Hooper, "The hyperledger project," in *Blockchain Enabled Applications*, pp. 139–149, Springer, 2017.

[21] "FISCO BCOS, a consortium blockchain platform," https://fisco-bcosdocumentation.readthedocs.io/en/latest/.

[22] Y. Qiu, Y. Liu, X. Li, and J. Chen, "A novel location privacy-preserving approach based on blockchain," *Sensors*, vol. 20, no. 12, p. 3519, 2020.

[23] D. Velicanu and C. Fromknecht, "CertCoin: a namecoin based decentralized authentication system," Technical Report, 6.857 Class Project, Massachusetts Institute of Technology, 2014.

[24] L. Axon and M. Goldsmith, "PB-PKI: a privacy-aware blockchain-based PKI," in *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, pp. 311–318, Madrid, Spain, 2017.

[25] S. Matsumoto and R. M. Reischuk, "IKP: turning a PKI around with decentralized automated incentives," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 410–426, San Jose, CA, USA, 2017.

[26] A. Hari and T. V. Lakshman, "The internet blockchain: a distributed, tamper-resistant transaction framework for the internet," in *HotNets '16: Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pp. 204–210, New York, NY, USA, 2016.

[27] J. G. Faisca and J. Q. Rogado, "Personal cloud interoperability," in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–3, Coimbra, Portugal, 2016.

[28] T.-T. Kuo and L. Ohno-Machado, "ModelChain: decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks," 2018, https://arxiv.org/abs/1802.01746.

[29] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: an authenticated data feed for smart contracts," in *CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 270–282, New York, NY, USA, 2016.

[30] M. Al-Bassam, "SCPKI: a smart contract-based PKI and identity system," in *BCC '17: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pp. 35–40, New York, NY, USA, 2017.

[31] S. Raju, S. Boddepalli, S. Gampa, Q. Yan, and J. S. Deogun, "Identity management using blockchain for cognitive cellular networks," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, Paris, France, 2017.

[32] T. Hardjono and A. Pentland, "Verifiable anonymous identities and access control in permissioned blockchains," 2019, https://arxiv.org/abs/1903.04584.

[33] J. Benet, "IPFS-content addressed, versioned, P2P file system," 2014, https://arxiv.org/abs/1407.3561.

[34] N. Kshetri, "Blockchain's roles in strengthening cybersecurity and protecting privacy," *Telecommunications Policy*, vol. 41, no. 10, pp. 1027–1038, 2017.

[35] M. Castro and B. Liskov, "Practical byzantine fault tolerance," *OSDI*, vol. 99, pp. 173–186, 1999.

[36] L. B. Martinkauppi, Q. He, and D. Ilie, "On the design and performance of Chinese OSCCA-approved cryptographic algorithms," in *2020 13th International Conference on Communications (COMM)*, pp. 119–124, Bucharest, Romania, 2020.

[37] H. Tschofenig and M. Pégourié-Gonnard, "Performance investigations," *IETF Proceeding*, vol. 92, 2015.

[38] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 313-314, Athens, Greece, 2013.