Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.045679

ARTICLE

A Cloud-Fog Enabled and Privacy-Preserving IoT Data Market Platform Based on Blockchain

Yurong Luo, Wei You*, Chao Shang, Xiongpeng Ren, Jin Cao and Hui Li

School of Cyber Engineering, State Key Laboratory of Integrated Service Network, Xidian University, Xi'an, 710126, China

*Corresponding Author: Wei You. Email: wyou@xidian.edu.cn

Received: 04 September 2023 Accepted: 29 November 2023 Published: 29 January 2024

ABSTRACT

The dynamic landscape of the Internet of Things (IoT) is set to revolutionize the pace of interaction among entities, ushering in a proliferation of applications characterized by heightened quality and diversity. Among the pivotal applications within the realm of IoT, as a significant example, the Smart Grid (SG) evolves into intricate networks of energy deployment marked by data integration. This evolution concurrently entails data interchange with other IoT entities. However, there are also several challenges including data-sharing overheads and the intricate establishment of trusted centers in the IoT ecosystem. In this paper, we introduce a hierarchical secure data-sharing platform empowered by cloud-fog integration. Furthermore, we propose a novel non-interactive zero-knowledge proof-based group authentication and key agreement protocol that supports one-to-many sharing sets of IoT data, especially SG data. The security formal verification tool shows that the proposed scheme can achieve mutual authentication and secure data sharing while protecting the privacy of data providers. Compared with previous IoT data sharing schemes, the proposed scheme has advantages in both computational and transmission efficiency, and has more superiority with the increasing volume of shared data or increasing number of participants.

KEYWORDS

IoT data sharing; zero-knowledge proof authentication; privacy preserving; blockchain

1 Introduction

More than 15 billion devices including smart grid sensors will build up a new IoT application model consisting of massive clusters of edge fog nodes and mega cloud centers in the next 10 years [1]. These devices are trending towards data processing and decision-making capabilities from sensing the environment. The rapid expansion of the IoT has also created an enormous quantity of multi-modal data [2]. Intelligent applications require deep integration and collaboration of multi-source data, specifically in the smart grid, events, such as power transmission strategy development, fault handling and billing. For example, accurate weather forecasts or environmental monitoring data are used to guide power transmission strategies [3]. The Smart Grid (SG) has established a wide-ranging and intelligent energy transmission network based on information flow. SGs are characterized by distributed management, intelligent control and two-way (communication & electricity) information

[4]. Sensors are involved throughout the service of the smart grid, and contemporary smart grids connect to the Internet of Things to bring people more comprehensive electrical applications and information services with higher quality. Meanwhile, the widespread deployment of power equipment in IoT-based SGs will generate large-scale and diversified energy data [5] that can support other IoT applications. Data sharing between smart grids and other IoT applications is extremely valuable, so effective management and secure sharing of data are the key secure requirements for SGs [6].

The smart grid is essentially an intelligent network consisting of massive sensors and electrical devices that link to the Internet. SGs achieve identification, positioning, monitoring, control and management, with a view to achieving the effect of interconnection of power networks and their customers [7]. The SG's two-way communication information is mainly derived from data collected by the IoT (internal and external of the SG power grid chain). Given that the IoT is a collaborative ecosystem and the huge data resources need to be used wisely and maximally, the secure sharing or trading of data is of extreme importance. At the same time, as an important part of the industrial Internet of Things, the smart grid data interaction and sharing with other IoT applications are the foundation of integrated applications. A large number of third-party data trading platforms have been established, but most of them are based on centralized architectures, whereby all data storage and transmission is done on a single central control system, which poses a challenge to trust establishment for the stakeholders of data sharing. At the same time, for power systems, large amounts of data are difficult to store and manage in a distributed manner [8]. Current third-party data trading platforms can provide storage of data to IoT devices through a cluster of servers in the cloud. However, as the number of devices and data increases, it will lead to a surge in transmission, an increase in latency and a decrease in sharing efficiency. Most of the data traded in third-party platforms are enterprise-level data, and there are often few transactions for the fragmented data held by individual users, which results in a shortage of diversified data sources. The introduction of fog nodes can promote sporadically distributed IoT devices with low computing power like SG devices for low-latency data sharing [9].

The increased scale of individual users is also very valuable for the smart grid which can contribute to the diversification and enrichment of data sets. However, individual users are reluctant to trade their data on a platform because the transaction process is cumbersome and the transmission costs and trust establishment far outweigh the benefits. Blockchain is a distributed database that enables secure, tamper-evident data storage in a non-centrally backed, non-secure environment [10]. In addition, it has the following advantages such as public verifiability, data traceability and tamper-proof, which can be a better solution for individual users' data transactions. Blockchain and cloud-fog-based architecture can provide a secure and convenient trading platform for individuals with low data amounts, and also facilitate the purchase of personal data by data buyers such as smart home manufacturers. Considering that malicious nodes can arbitrarily access data resources, it is a key point to achieve entity authentication in data processing. In addition, depending on data security requirements, data providers prefer to authenticate the data purchaser to know their true identity [11]. Because data providers have to analyze the purchaser's intention and the subsequent processing of the data. Also, they need to define the scope of data use after ownership has been transferred to avoid the misuse of those data. European Union (EU) and China data protection regulations explicitly require identity authentication of the data controller (i.e., the data consumer in a data transaction). Art.13 of the EU General Data Protection Regulation (GDPR) [12] states that the identity of the data collector should be provided to the data owner. Current blockchain-based data-sharing solutions focus on solving technical problems such as bidding for transactions and transferring data ownership, and cannot

achieve entity authentication, which may result in data misuse remaining untraceable after the transfer of data even with the use of blockchain.

In this paper, we propose and implement a privacy-preserving decentralized secure data sharing platform named **Zobric** for smart grid and IoT devices. The main innovations and contributions of this paper are as follows:

(1) We design a cloud-fog enabled and privacy-preserving hierarchical data sharing model and introduce Hyperledger Fabric technology to build a identity management and transaction architecture for IoT nodes with low power consumption. The platform adopts a scheme where fog nodes collect, encrypt and publish data from the same management domain, and cloud servers act as transit points to store data. The proposed scheme can solve the problems of data encryption and transmission difficulties caused by limited computing resources.

(2) We propose a non-interactive zero-knowledge proof-based group authentication and key agreement protocol that supports batch data sharing transactions and one-to-many sharing sets of data. The true identity of the data provider can only be known to the data consumer, which effectively protects the privacy of data providers. A transaction grouping method is adopted in which data is classified and tagged based on the source. Platform users join transaction groups according to the type of data they are sharing, using the Fabric channel to achieve a private transaction channel and a low blockchain storage consumption. The solution improves the efficiency of data transactions and enables the accurate transfer of data ownership with privacy preserving and untraceability.

(3) We develop and realize a series of smart contracts to autonomously build, update and control cloud server data access lists to enable the reliable flow and efficient sharing of data collected from large-scale distributed and resource-constrained devices.

(4) We employ a formalization security analysis tools and demonstrate that our proposed solution can resist against various protocol attacks and achieve several security properties including authenticity and confidentiality. And we implement experimental performance simulations to validate that the proposed IoT/SG data-sharing architecture enables the efficient and secure exchange of sensory data, which is more suitable for devices at scale, across domains and with low computing power.

The structure of this paper is as follows. We introduce related work in Section 2 and explain some background knowledge in Section 3. Section 4 describes the system and the security model. In Section 5, the proposed scheme is described in detail. We give the security proof and performance analysis of the scheme in Section 6 and Section 7, respectively. Finally, Section 8 concludes the article.

2 Related Work

The data sharing/trading platforms currently in use are mainly hosted by large corporations or national governments and all employ a centralized trading architecture, such as Factual Open Platform [13] for location data and services. Such centralized trading platforms cannot provide data marketing for individual users between different IoT applications because the cost of monitoring the transfer of data ownership is extremely high. In addition, centralized platforms have access to the plaintext of data and cannot prevent platforms from misusing data or eavesdropping on users' privacy information [14].

Access control mechanisms are the general security solution for data distribution, so control models with a third-party authorization are proposed for IoT data security [15,16]. Zhang et al. [17]

proposed an access control scheme using Ciphertext-Policy Attribute-Based Encryption (CP-ABE) in fog computing which realizes data sharing and attribute revocation but not trading for individuals. Data sharing is generally achieved using ciphertext and attribute-based encryption technology [18] on untrusted cloud storage servers which require a large computational overhead due to data processing. Sun et al. [19] designed a privacy-preserving data management model based on edge service layer and attribute-based encryption. Tan et al. [20] proposed a blockain-enabled access control framework for green IoT smart devices.The schemes based on CP-ABE can achieve one-to-many public cloud data sharing, but they cannot simultaneously share multiple pieces of data. Moreover, the above solutions do not take into account the identity management issues of both parties to the data exchange.

There is a category of blockchain-based IoT/SG data sharing platforms designed for trading bidding solutions [21,22]. They deployed smart contracts to make auctions for crowdsourcing data trading. Li et al. [23] proposed a framework of blockchain-enhanced decentralized IoT data trading, which cannot provide the secure solution for data sharing. Dixit et al. [24] presented a marketplace model and architecture to support trading of streaming data and yet focused on deal negotiation. Yu et al. [25] proposed a blockchain-enabled security data access control scheme. Boo et al. [26] introduced zero-knowledge proof protocol and designed a blockchain-based IoT data trading framework which supports traditional one-to-one data transmission, but cannot authenticate the transaction entities. Özyilmaz et al. [27] proposed a distributed marketplace for central-free machine learning data trading. However, the scheme does not protect the shared data with encryption. Chen et al. [28] developed a data batch aggregation mechanism for smart grids based on double-blockchain which introduces more consensus overhead. Zhang et al. [29] proposed a privacy-preserving and user-defined data sharing architecture with fine-grained access policy, utilizing blockchain technology and attribute-based cryptosystem. Walshe et al. [30] proposed a non-interactive zero-knowledge proof (NIZKP) authentication protocol for IoT devices with multi-rounds of challenges and high computational overhead. However, none of the above schemes enable the authentication of both parties to a transaction. Table 1 shows the properties comparison between the proposed scheme and previous related schemes.

Table 1: Comparison with previous related schemes

Property	[14]	[16]	[19]	[23]	[24]	[25]	[28]	[30]	Ours
Distributed management	✓	✓	×	✓	✓	✓	×	✓	✓
Blockchain type (Consortium/Private)	C	–	–	P	C	C	C	–	C
Data access control	✓	✓	✓	✓	✓	✓	×	×	✓
Privacy preserving	✓	✓	✓	×	✓	×	✓	✓	✓
Group sharing	×	✓	✓	×	×	×	✓	×	✓
Identity authentication	×	×	×	×	✓	×	×	✓	✓
Data storage location (Cloud/Devices)	C	C	D	D	D	C	C	–	C

3 Preliminaries

3.1 Non-Interactive Zero-Knowledge Arguments [31]

3.1.1 Quadratic Arithmetic Program (QAP) Generator

For a finite field $\mathbb{F}$ and integers $1 \leqslant l \leqslant m$, a QAP relation R is a set of binary pairs (ϕ, ω) in which $\phi = (a_1, \ldots, a_l) \in \mathbb{F}^l$ is the statement and $\omega = (a_{l+1}, \ldots, a_m) \in \mathbb{F}^{m-l}$. For $a_0 = 1$, let $u_i(X)$, $v_i(X)$ and $w_i(X)$ be degree $n-1$ polynomials, ϕ and ω satisfy $\sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) \equiv \sum_{i=0}^{m} a_i w_i(X) \bmod t(X)$.

3.1.2 Zero-Knowledge Arguments

A publicly verifiable zero-knowledge argument for relation R is a set of probabilistic polynomial algorithms **(Setup, Prove, Vfy, Sim)** as follows:

Setup $(R, \lambda) \rightarrow (\sigma, \tau)$: Setup derives a Common Reference String (CRS) σ as a public parameter and a simulation trapdoor τ for the relation R.

Prove $(R, \sigma, \phi, \omega) \rightarrow \pi$: Prover porduces an argument π with inputing σ and $(\phi, \omega) \in R$.

Vfy $(R, \sigma, \phi, \pi) \rightarrow \{0, 1\}$: Verifier enters the CRS σ, the ϕ and the argument π as input, and verification returns 0 for reject or 1 for accept.

Sim $(R, \tau, \phi) \rightarrow \pi$: Simulator produces an argument π with inputs the trapdoor τ and the statement ϕ.

Non-interactive zero-knowledge algorithms in [31] has been proven to have COMPLETENESS, ZERO-KNOWLEDGE, and COMPUTATIONAL KNOWLEDGE SOUNDNESS security capabilities defined as follows.

COMPLETENESS. Completeness denotes that, a trustworthy prover can persuade verifiers who correctly execute **Vfy** with the true statements. That is, $Pr[(\sigma, \tau) \leftarrow \textbf{Setup}(R); \pi \leftarrow \textbf{Prove}(R, \sigma, \phi, \omega) : \textbf{Vfy}(R, \sigma, \phi, \pi) = 1] \approx 1$.

ZERO-KNOWLEDGE. Zero-knowledge denotes that, public parameters, including statement, will not disclose confidential information. That is, for any polynomial time adversaries $\mathscr{A}$, $Pr[(\sigma, \tau) \leftarrow \textbf{Setup}(R); \pi \leftarrow \textbf{Prove}\ (R, \sigma, \phi, \omega) : \mathscr{A}(R, z, \sigma, \tau, \pi) = 1] - Pr[(\sigma, \tau) \leftarrow \textbf{Setup}(R); \pi \leftarrow \textbf{Sim}\ (R, \tau, \phi) : \mathscr{A}(R, z, \sigma, \tau, \pi) = 1] \approx 0$.

COMPUTATIONAL KNOWLEDGE SOUNDNESS. Soundness denoted that, provers cannot prove to verfier without witness, nor can they generate false statement to pass **Vfy**. That is, for any polynomial time adversaries, $Pr[(\sigma, \tau) \leftarrow \textbf{Setup}(R); (\phi, \pi) \leftarrow \mathscr{A}(R, z, \sigma) : \phi \notin L_R$ and $\textbf{Vfy}(R, \sigma, \phi, \pi) = 1] \approx 0$. Strengthening soundness to knowledge soundness, if there exists an extractor which can generate a witness every time that the adversary produces a valid argument, it know all about the adversary's state including any random coins. Formally, for any polynomial time adversaries $\mathscr{A}$ with a non-uniform polynomial time extractor $\mathscr{X}_{\mathscr{A}}$ such that $Pr[(\sigma, \tau) \leftarrow \textbf{Setup}(R); ((\phi, \pi); \omega) \leftarrow (\mathscr{A} || \mathscr{X}_{\mathscr{A}})(R, z, \sigma) : (\phi, \omega) \notin R\ ((\phi, \pi); \omega) \leftarrow (\mathscr{A} || \mathscr{X}_{\mathscr{A}})(R, z, \sigma) : (\phi, \omega) \notin R$ and $\textbf{Vfy}(R, \sigma, \phi, \pi) = 1] \approx 0$.

3.2 Elliptic Curve Discrete Logarithm Problem [32]

An elliptic curve G over the field of integers modulo q $\mathbb{F}_q$ is a point set $(x, y) \in \left(\mathbb{F}_q\right)^2 \cup 0$, where 0 is the point at infinity. And the point can satisfy the equation $y^2 \equiv x^3 + ax + b \bmod q$ where $4a^3 + 27b^2 \neq 0 \bmod q$. For algebraic sum and scalar multiplication over group G and a base point P, there is a computational hard prolem ECDLP without known polynomial time algorithm to solve it. The problem is defined as follows:

Elliptic Curve Discrete Log Problem (ECDLP): Given P and a point $Q \in G$, where $Q = zP$, find z from Z_q^*.

4 System and Security Model

4.1 System Model of Zobric

Based on the typical system architecture in IEEE IoT data management standards [33], the system architecture of the proposed IoT data market platform is shown in Fig. 1, which consists of multiple parties. The following three roles of stakeholders in the proposed platform are defined.

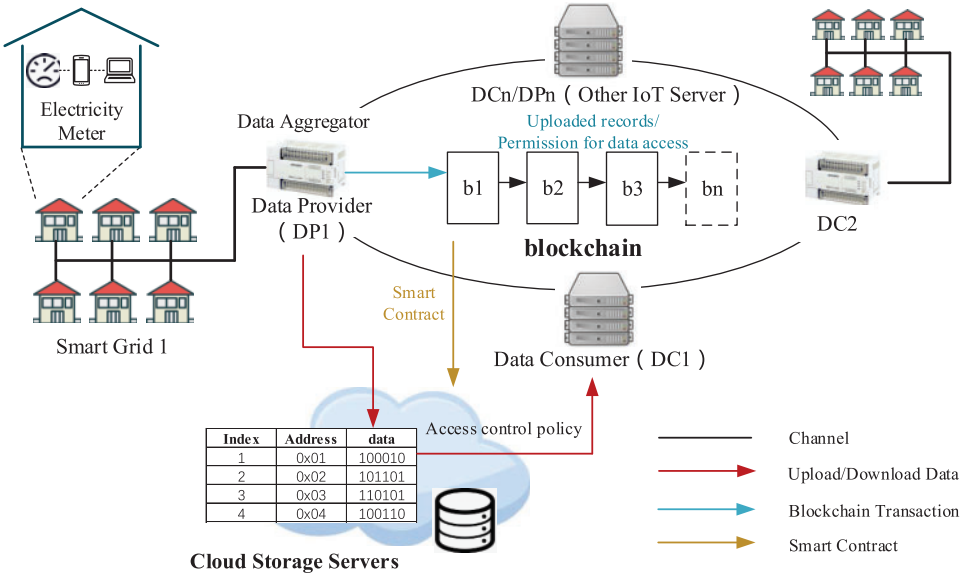

Figure 1: Cloud-fog enabled IoT data market platform

Data Provider (DP). Data provider is defined as a party owning the data and wishing to sell or share it. The DP consists of a cluster of IoT sensors, which collect various types of data, and a cluster control center with high computing/storage/information transmission capability, which manages and controls the group devices. In this architecture, data transaction activities are carried out by the Data Control Center on behalf of the Data Provider.

Data Consumer (DC). Data consumer is defined as a party who wishes to purchase or receive data from Data provider. Generally, data consumers need to purchase data in their real identity and reach a data use agreement with the data provider in order to be granted ownership of the data.

Data Trading Services Platform. The service platform is a system for data trading. It consists of two entities: the Cloud Server (CS) and the Blockchain Server (BS) nodes, which provide infrastructure services for data providers and data consumers. For HyperLedger Fabric, BS nodes include CA, Orderers, Endorsers, Comitters and Anchors.

4.2 Adversary Model and Assumption

Without loss of generality, we set up the Dolev-Yao intruder model [34] as a security model within the IoT data trading platform. Adversaries in this model have access to all messages passing over the public network and may be legitimate entities or pretend to be legitimate users who can initiate sessions with or receive messages from any entity within the system. But an adversary is also limited

in his ability to extract the private key through public key cracking, or to obtain information from the private storage portion of other entities. In addition, the following security assumptions are applied to the trust relationships of the system entities.

1. **Threat model of entities.** External attackers have the same capabilities as described in the Dolev-Yao model. The internal potential adversaries are defined as follows. Individual users of the platform may try to disguise, replay and sell fake, invalid or duplicate data, at the same time, data buyers may misuse of shared data. Cloud storage servers are assumed to be honest-but-curious, specifically, the cloud servers execute the contents of smart contracts correctly but also attempt to view the data it stores.
2. **Blockchain security.** Data stored on the blockchain is public to all nodes. Consortium blockchain peers are authorized to register to the platform but there might be malicious nodes. Edge nodes (Data providers and consumers) are endorsors for the blockchain and realize consensus based on Practical Byzantine Fault Tolerance (PBFT) [35].
3. **Consensus fault tolerance.** Invalid nodes in fabric refer to nodes that cannot perform validity verification due to downtime or malicious operations. From the PBFT principle, the amount of invalid nodes (including malicious nodes) in the system must not exceed 1/3 of the committee validator nodes.

4.3 Security Goals

The Zobric platform is designed with the following security goals.

1. **Mutual Authentication.** Data providers need to authenticate data consumers and clarify the flow of data for the sake of traceability of subsequent data processing. In addition, data providers need to achieve mutual authentication with data consumers under the premise of lacking an authentication center.
2. **Secure One-to-Many Data Sharing.** If a data provider sells several sets of data to multiple data consumers using one-to-one data sharing schemes, it will result in a significant increase in transaction volume and computational cost of data provider in a distributed system. In addition, as cloud servers or data trading platforms are untrusted, the platform should also support secure sharing among data providers in bulk in the IoT environment.
3. **Privacy Preserving.** Data providers are unwilling to disclose the trading or sharing status of the data. Meanwhile, data plaintext should not be accessed by unauthorized entities, including cloud servers. So the identity of the data provider should be secret during the transaction process, meanwhile the data consumer can be associated with the data provider.
4. **Data Access Control.** Data transactions should guarantee that only authenticated data consumers who are authorized for the data have access to the specified data. This setting should be controlled by the data provider, but could be operated by the cloud server. Data providers should dynamically adjust the policy for data access control in real-time based on the transaction.
5. **Traceability, Auditable and Integrity of Data Transaction Records.** In an IoT environment without a trusted center, data trading platforms need to record data transactions information for accountability including the entities of the transaction, data type, data volume, etc. In order to provide post-transaction fairness and negotiability, data transaction records should be guaranteed to be traceable and tamper-resistant upon consensus. In addition, records of data ownership transfer should be auditable.

6. **Resisting against Multiple Typical Attacks.** The proposed authentication and data sharing protocols should be executable and resistant to existing protocol attacks such as replay attacks, Man-in-the-middle attacks, forgery attacks, data stealing and so on.

5 Zobric Design

The proposed IoT Data Market Platform Zobric runs on 3 layers: (1) The bottom layer is the data aggregation layer where data collection is performed by various IoT terminal devices; (2) The middle layer is the data control and transmission layer, in which the device group control center performs data sharing control and the establishment of secure channels for data transmission among IoT devices; (3) The top layer is the sharing and trading layer, where the data is traded or shared among the data provider, the demander and the cloud storage server, with secure authentication and key agreement protocol and smart contracts. The secure data transaction procedures are shown in Fig. 2 and described as follows in detail.

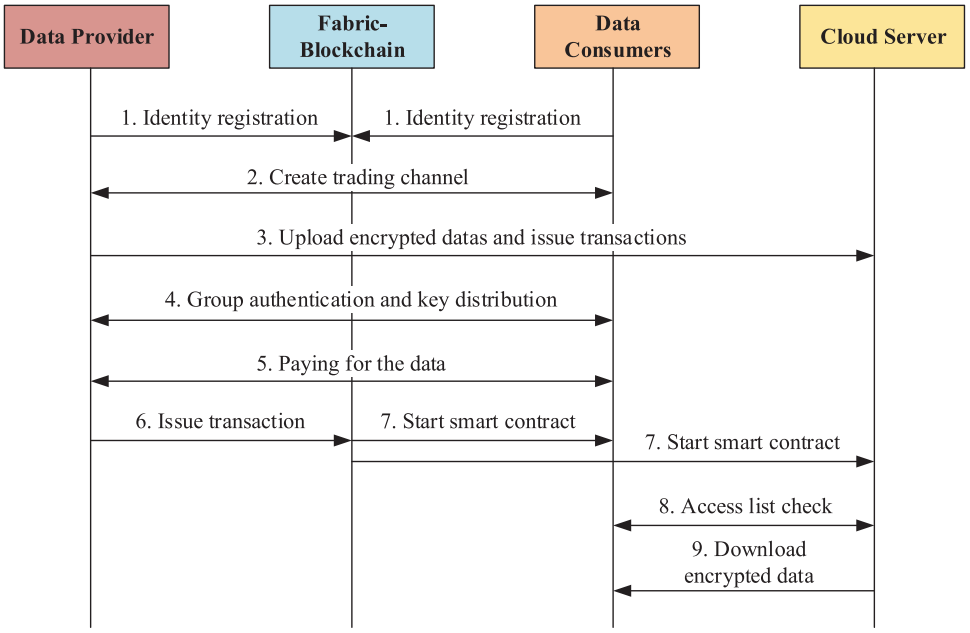

Figure 2: Procedures for zobric data sharing

1) Identity registration, where the nodes involved in data transactions register in the data trading platform with the help of BS.
2) Transaction group creation, where a data provider issues a data announcement or a data consumer initiates a data request to create a transaction group (Fabric Channel) to protect transaction privacy.
3) The data provider encrypts the data using symmetric encryption with a data protection key and uploads it to the cloud server side. This step can be repeated multiple times for multiple sets of data. Then DP will issue data uploading transactions and wait for transaction confirmation.
4) The group authentication and key agreement protocol would be executed among the data provider and the data consumers to negotiate data protection keys.
5) The data provider and data consumers negotiate data bids/payments, etc.

6) The data provider summarizes the valid data trades mentioned above, and only one blockchain data authorization transaction is issued, (with a list of storage addresses created by the cloud server) to authorize the data for the data consumers.
7) Once the data authorization transaction is confirmed, a smart contract is triggered to create/update a list of data access rights on the cloud server and the data consumers side.
8) The cloud server control the access of the consumers based on list of permissions.
9) The authenticated data consumer can download the data and use the data encryption key to decrypt the data.

5.1 Group Authentication and Key Agreement Protocol

5.1.1 Registration and Setting Up Phase

The notations used in the protocol and their corresponding definitions are shown in Table 2. On the registration phase, DPs and DCs need to register in the Fabric-based consortium chain to join the trading platform. The blockchain consists of identity management nodes and Fabric-CA. In this scheme, the node identity registration is done based on the existing registration architecture, and the registration process will be done under TLS (Transport Layer Security) protection. The registration procedures are as follows.

Table 2: Definition of notation

Notation	Definition
ID_i	Identity of node i
DI_j	Index of data j
DA	Data address
$\mathbb{G}_i$	Elliptic curve
s_i	Secret of entity i
(PK_i, SK_i)	Public/Private key pair of entity i
(APK_i, ASK_i)	Aggregated signature key pair of node DC_i
$Cert_i$	Certificate of data transaction node i
GS	Data encryption key
V_{pub}	Transaction content public declaration, ie. uploaded or authorized data DI_j
V_{trans}	Transaction content declaration after signature with SK_{DP_i}
ω_i	Verification string
$Enc_{SK}(\cdot)$ / $Dec_{SK}(\cdot)$	Symmetric encryption/decryption using key SK
$Sig_{SK}(\cdot)$ / $Ver_{PK}(\cdot)$	Signature/Verification using key SK/PK

1. The data transaction node i that wishes to join the platform sends a registration request $(i - specific, s_i)$ to the blockchain service node, $i - specific$ means that the real identity of node i, such as the combination of the username and its transaction identity, s_i is the secret of node i, such as the password.
2. After receiving the request message, the platform verifies the identity with consortium policy and generates an unprecedented identity ID_i for node i, and the blockchain service node

generates the public/private key pair (PK_i, SK_i) and certificate $Cert_i$, then replies $((PK_i, SK_i), Cert_i)$ to the node.

Blockchain service nodes (BSs) will set public parameters for transaction validation and authentication initialization on the setting up phase.

(1) The BS in the trading platform performs authentication initialization, selects the security parameter $k \in Z^*$, and generates the authentication system parameters as follows:

(1.1) Choose an elliptic curve $\mathbb{G}_1$ of prime order q. Assuming that the generator of the curve is P. Choose the master key $s_{BS} \in Z_q^*$, the public key is $P_{BS} = s_{BS}P$.

(1.2) Choose three hash functions as follows, $H_1 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \to Z_q^*$, $H_2 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_1 \times \{0,1\}^* \times \mathbb{G}_1 \to Z_q^*$, $H_3 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_1 \times \{0,1\}^* \to Z_q^*$, and expose public parameters $\{\mathbb{G}_1, q, P, P_{BS}, H_1, H_2, H_3\}$.

(1.3) Choose a nonce $r_i \in Z_q^*$, compute $R_i = r_iP$ and $h_{1i} = H_1(ID_{DC_i}, R_i, P_{BS})$. Then compute a partial private key $k_i = (r_i + s_{BS}h_{1i}) \bmod q$, and send $K_{1i} = (k_i, R_i)$ to the DC_i through a secure channel.

(2) The DC_i generates the aggregated signature key pair (APK_i, ASK_i) as follows:

(2.1) After receiving the K_{1i}, check whether the equation $k_iP = R_i + h_{1i}P_{BS}$ holds, if it holds, go to the next step.

(2.2) Generate a random number as s_{DC_i} and compute $K_{2i} = s_{DC_i}P$, then set the public key $APK_i = (K_{2i}, R_i)$ and the private key $ASK_i = (s_{DC_i}, k_i)$.

(3) The BS generates the zero-knowledge proof authentication parameters for the DP as follows:

(3.1) Invoke $NIZK(\sigma, \tau) \leftarrow Setup(R)$ as described in Section 3, in which the calculation constraints for relation R are as follows, they are a series of polynomials that form computational problems.

Balance: Assumed that this transaction includes $m - l - 1$ pieces of data, for each data $DI_j (l + 2 \leqslant j \leq m)$ in data list DL to be traded: $\sum_{j=l+2}^{m} V_{pub_j}^{upload} + \sum_{j=l+2}^{m} V_{trans_j}^{upload} = V_{pub}^{trade} + \sum_{j=1}^{m} V_{trans_j}^{trade}$.

Trade authority: For uploading transaction block serial number for DI_j (generally, privacy protection is performed) should be equal to trade transaction block which points out the upload block serial number.

Data ownership authentication: For each $j (l+2 \leqslant j \leq m)$ and a Hash function $H_4 : \{0,1\} \to Z_q^*$, satisfy the following equation.

$$Ver_{PK_{DP}}(Sig_{SK_{DP}}(H_4(DI_j, GS_{DI_j}))) = 1$$

(3.2) For each witnesses $(a_{l+1}, \ldots, a_m) \in Z_P^{m-l}$, $a_{l+1} = SK_{DC}$ $a_j = GS_{DI_j}, l + 2 \leq j \leq m$, and $a_0 = 0$, statment $(a_1, \ldots, a_l) \in Z_P^l$ can be determined based on R. $V_{pub_j} = DI_j$ and $V_{trans_j} = Sig_{PK_{DP}}(DI_j)$. The equation holds relation R: $\sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) = \sum_{i=0}^{m} a_i w_i(X) + h(X)t(X)$, in which $t(X)$ is an $(n-2)$th degree polynomial.

(3.3) Assuming G, H is the arbitrary generator of group $\mathbb{G}_2$ and $\mathbb{G}_3$, respectively. The DC chooses a random nonce $\in Z_P^*$, and the BS chooses random nonces $\alpha, \beta, \gamma, \delta, x \in Z_P^*$. Set $\tau = (\alpha, \beta, \gamma, x)$ and compute the public verification parameter common reference string σ

according to the following equation:

$$\sigma = \left\{ \begin{array}{c} G^{\alpha}, G^{\beta}, H^{\beta}, H^{\gamma}, G^{\delta}, H^{\delta}, \left\{G^{x^i}\right\}_{i=0}^{n-1}, \left\{H^{x^i}\right\}_{i=0}^{n-1} \\ \left\{G^{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma}}\right\}_{i=0}^{l}, \left\{G^{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}}\right\}_{i=l+1}^{m}, \left\{G^{\frac{x^i t(x)}{\delta}}\right\}_{i=0}^{n-2} \end{array} \right\}$$

5.1.2 Trading Group Formation and Negotiation

1. For registered users, the corresponding data trading channel can be connected according to the demand.
2. The data provider broadcasts the data sets information in the channel, such as data size, content summary, data price, etc. The data consumers $(1-i)$ wishing to purchase this data join the group for this transaction and execute the group authentication and key agreement protocol in 5.1.4 with the data provider.

5.1.3 Data Uploading Transaction

Data providers can select the datasets to be uploaded depending on consumer feedback in the trading channel, and this operation increases the probability that the data will be purchased. For a data provider DP, it generates $DI = H(Data)$ and storage encryption GS for one set of data, then it applies to upload the encrypted data to the cloud. The cloud server will check if the DI appears repeatedly, if not, it will accept the request and assign a storage address DA to DI. After the upload is completed, the DP will initiate an upload transaction with $(DI, Sig_{PK}(DI), Sig_{PK}(H(DI, GS)))$. The DP can repeat this operation by uploading multiple sets of data for sale. It should be noted that data trading can only be carried out after the uploading transaction confirmation.

5.1.4 Group Authentication and Key Agreement

As shown in Fig. 3, the authentication and key agreement procedures are as follows:

(1) At first, the DP broadcasts a list of tradable data $DI_l + 1...Dn$, $m \leq n$ within the trading channel, and provides the name $TN = (TradingNumber)$ and valid time T_V of this trading. Note that during this period, the DP does not need to send its real ID and can use a temporary ID to achieve trading anonymity.

(2) Within the validity period, all DC_i can choose the data they want to buy and generate a purchase list $PL_i = (DI_j...)$. Then the DC_i chooses a random nonce $y_i \in Z_q^*$ and a timestamp T_{1i}, and computes the signature on $H(TN)$ according to the following equations: $Y_i = y_i P$, $h_{2i} = H_2(H(TN), ID_{DC_i} \parallel PL_i, APK_i, T_{1i}, Y_i)$, $h_{3i} = H_3(H(TN), ID_{DC_i} \parallel PL_i, APK_i, T_{1i})$, $W_i = (h_{2i} y_i + h_{3i}(s_{DC_i} + k_i)) \bmod q$. The DC_i replies the message $M_{1i} = (ID_{DC_i} \parallel PL_i, Y_i, W_i, T_{1i})$ to the DP.

(3) After receiving different response messages M_{1i} $(i = 1, \cdots, N)$ before T_V, the DP aggregates these messages by computing these equations: $h_{1i} = H_1(ID_{DC_i} \parallel PL_i, R_i, P_{BS})$, $h_{2i} = H_2(H(\pi), ID_{DC_i} \parallel PL_i, APK_i, T_{1i}, Y_i)$, $h_{3i} = H_3(H(\pi), ID_{DC_i} \parallel PL_i, APK_i, T_{1i})$. Then the DP computes the aggregated signature $U = \sum_{i=1}^{N} h_{2i}\ Y_i$, $W = \sum_{i=1}^{N} W_i$, and verifies whether the equation $WP - U = \sum_{i=1}^{N} h_{3i}(K_{2i} + R_i + h_{1i} P_{BS})$ holds, if it holds, these DC_is are valid, otherwise, the DP detects every member according to the equation $W_i P - h_{2i} Y_i = h_{3i}(K_{2i} + R_i + h_{1i} P_{BS})$, if it holds, the member and its PL_i is valid.

(4) The DP calculates proven data list $DL = (DI_{l+2}...DI_m) = \cup_{i=1}^{N} PL_i$. Then the DP chooses two nonces $r, s \in Z_P$, then invokes *Prove* to calculate Data ownership proof $\pi = (A, B, C)$, $A = G^{\alpha+\sum_{i=0}^{m} a_i u_i(x)+r\delta}$, $B = H^{\beta+\sum_{i=0}^{m} a_i v_i(x)+s\delta}$, $C = G^{\frac{\sum_{i=l+1}^{m} a_i(\beta u_i(x)+\alpha v_i(x)+w_i(x))+h(x)t(x)}{\delta}} + s(\alpha+\sum_{i=0}^{m} a_i\, u_i(x))+r(\beta+\sum_{i=0}^{m} a_i v_i(x)) - rs\delta$. The DP broadcasts the proof $M_2 = (\pi, DL, T_2)$ to all data consumers DC_i $(i = 1, \cdots, N)$ who need to purchase the data.

(5) After receiving the message, all DC_i first check T_2 freshness and can choose to invoke *Verify* to check whether the equation $e(A, B) = e(G^{\alpha}, H^{\beta}) e\left(G^{\sum_{i=0}^{l} a_i\left(\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma}\right)}, H^{\gamma}\right)$ $e(C, H^{\delta})$ holds. If it holds, the message is valid. Each DC_i should pay for the data they subscribe to.

(6) After confirming payment, the DP sends the message $M_{3i} = Enc_{PK_{DC_i}}(GS_{PL_i}, DA_{PL_i}, TN, ID_{DC_i})$ to those DC_is who have successfully purchased data and been authenticated. For the data DI_i sold to $DC_k..DC_j$, the DP issues a transaction within $Sig_{PK_{DP}}(List(DI_i, DC_k...D_j), DI_i \in DL)$ which indicates the ownership transfer list of every data DI_i in DL.

(7) After receiving the message M_{3i}, respective DC_i decrypts the M_{3i} to obtain the GS_{DI_j} it wants. The hash header in the transaction can be indexed to the data provider. Alternatively, the DC can directly try downloading data from the cloud.

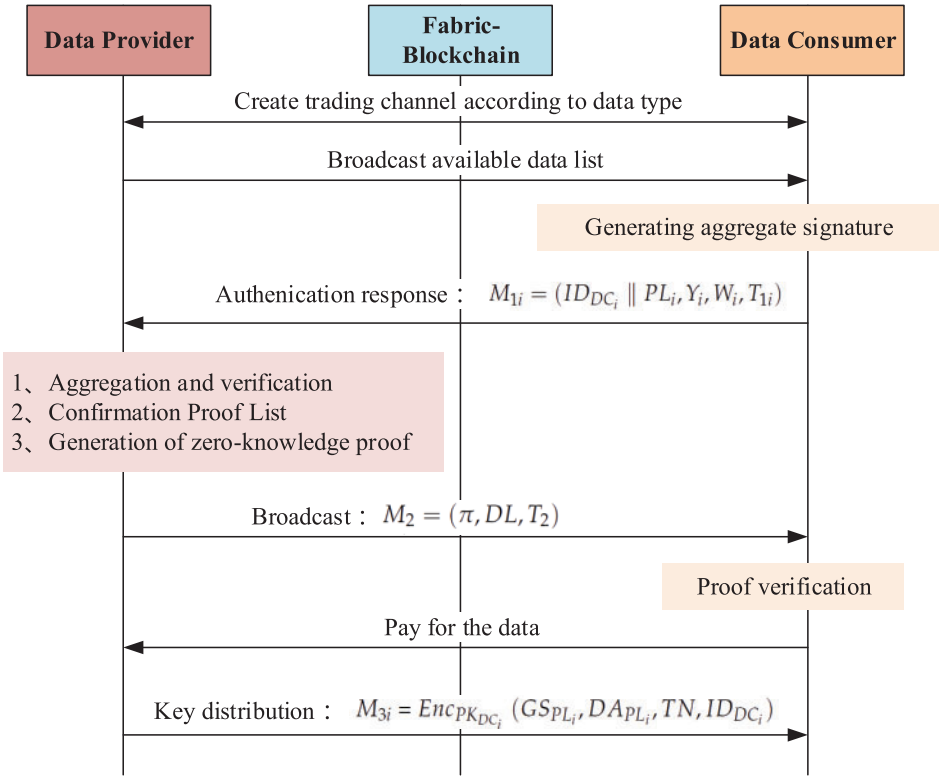

Figure 3: Group authentication and key agreement protocol

5.2 Fabric Smart Contract ZobricCC Design

The data provider publishes a transaction to record the trading. Specifically, the transaction provides data access lists in *DL* and the accessible data addresses.

5.2.1 Transaction Block Design

The structure of the published transaction block is shown in Fig. 4. **ChannelHeader** includes the transaction ID (Txid), timestamp (Timestamp), and channel information (ChannelId). **SignatureHeader** contains the certificate of the block creator, his ID (mspid) and a nonce. The header also contains the period information used to identify the logical time window that is used to resist replay attacks. **ChaincodeSpec** contains the version, name and timeout of the smart contract. **ProposalResponse** records the response to a smart contract call. **ChaincodeEndorseAction** contains the Endorser name and its Signature.

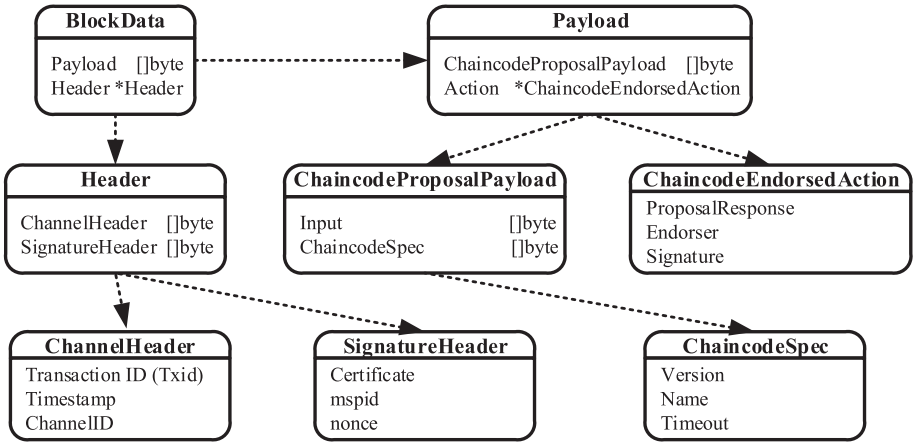

Figure 4: Transaction block structure

Input is the function of the proposed smart contract including the transaction, the data provider signature (DPSign), the address of the data on the cloud server (DA), and the data consumer ID list (DCs). Transaction amount (TAmount), and this part are specified by the smart contract. When updating the access list, **Input** contains the UpdateAccessList (function name), the IP address of the cloud server, the user name (DC) to be updated and the data address to be added (NewDA). The content of the **Input** is specified by the Zobric smart contract (ZobricCC). Fig. 5 shows the contents of each part when recording transactions and updating the permission list.

5.2.2 Smart Contract Functional Design

After the authentication procedures are finished, the smart contract is invoked for authorization. When a Smart Contract is executed, the Invoke method is first called, then the function of the Invoke and the parameters are analyzed before the relevant process is executed. The Invoke flows are shown in Fig. 6.

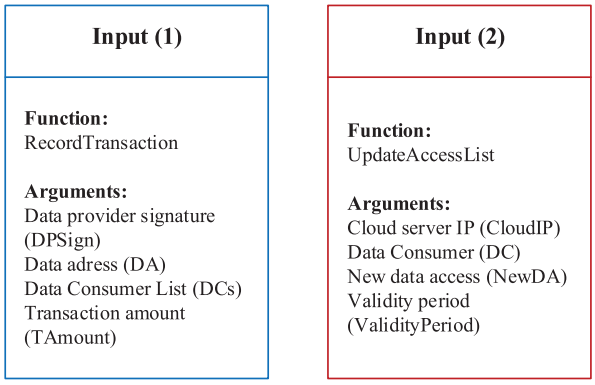

Figure 5: Contents of input function

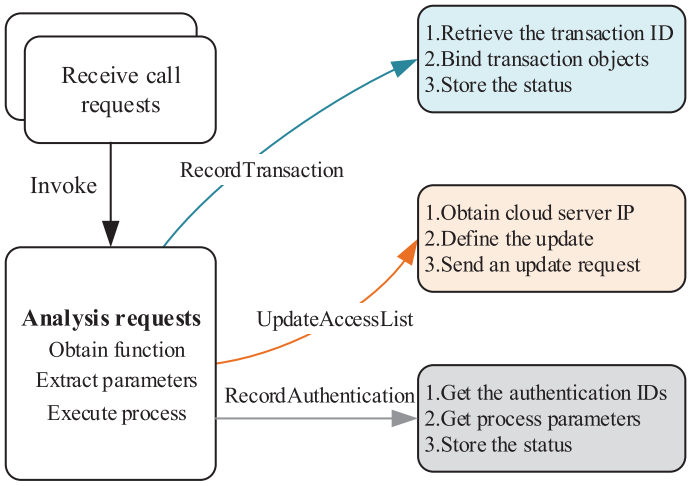

Figure 6: Smart contract invocation flow

(1) **RecordTransaction.** When a request is received to record a transaction, the transaction ID (TxId) in the ChannelHeader is first checked for uniqueness and a transaction record can be identified based on this ID. The request parameters are then bound to an object called transaction, which mainly contains the data provider signature, data address, data consumers, transaction amount and other information as shown in Fig. 5 **Input(1)**. The transaction is then checked for reasonableness, mainly to detect the existence of the user, and whether there are duplicate purchases. The TxId and transaction information is stored as a key-value pair in the state database after the test is completed and recorded in the chain.

(2) **UpdateAccessList.** After the transaction has been recorded, the access list on the cloud server also needs to be updated so that the consumer who has purchased the data has permission to obtain the data address. The smart contract receives a call with the function UpdateAccessList. After parsing the parameters, it first checks the permissions and the validity of the user and the transaction by querying the transaction information related to that user with the function queryByDC(). The UpdateRequest() method is then called to construct an update request, serialising the parameters, such as the user to be updated, the permissions to be added, and the

validity of the permissions. The cloud server receives the request and updates the permission lists.

(3) **RecordAuthentication.** Before a transaction is issued, the authentication and key agreement protocol needs to be completed, while protecting the privacy of data providers. The structure of ZKP is defined in the smart contract, which includes the transaction ID and the proof π. Similar to **RecordTransaction**, this function detects whether the transaction ID is unique, then binds the parameter to a ZKP object and stores the transaction ID and ZKP as a key-value pair in the state database. When the DC needs to use the parameter, it can query the parameter by calling queryProof (TransactionID).

6 Security Analysis and Discussion

6.1 Correctness Analysis

(1) **Mutual authentication.** The data provider authenticates all data consumers by verifying the aggregated signatures. The correctness of the signature algorithm is verified as follows:

$$\begin{aligned} W_i P - h_{2i} Y_i &= \left[h_{2i} y_i + h_{3i}\left(s_{DC_i} + k_i\right)\right] P - h_{2i} Y_i \\ &= h_{2i} y_i P + h_{3i}\left(s_{DC_i} P + k_i P\right) - h_{2i} Y_i \\ &= h_{3i}\left(k_{2i} + r_i P + s_{DC} h_{1i} P\right) \\ &= h_{3i}\left(k_{2i} + R_i + h_{1i} P_{BS}\right) \end{aligned}$$

The correctness of the aggregation algorithm is verified as follows:

$$\begin{aligned} WP - U &= \left(\sum_{i=1}^{N} W_i\right) P - \sum_{i=1}^{N} h_{2i} Y_i \\ &= \sum_{i=1}^{N} \left(W_i P - h_{2i} Y_i\right) \\ &= \sum_{i=1}^{N} h_{3i}\left(k_{2i} + R_i + h_{1i} P_{BS}\right) \end{aligned}$$

The aggregated signature algorithm is based on the ECDLP, and the proof of security can be found in [36]. The data consumers signs the fresh identity proof TN with secret key s_{DC_i} to respond to the authentication challenge. Moreover, zero-knowledge arguments in the proposed protocol have been proven to be computational knowledge soundness, that is, the DC who has $SK_D C$ and knows the GS_{PL} can construct the convinced proof π. This proves the identity of the DC and its ownship of all data in the data list DL.

(2) **Key distribution and secure data sharing.** The scheme can ensure the security of data sharing. Firstly, the DP authenticates the data consumers and verifies the applicant for their required data list. Secondly, the DP encrypts the data key using the public key of data consumer, which ensures that only authenticated and payment confirmed consumers can obtain the key. Thirdly, the DP releases the data access list specified in the transaction and triggers the smart contract to update the data access permission list on the cloud server side. Even if the cloud server is untrustworthy or there exists disguised attackers, the data sharing security can be guaranteed.

(3) **Privacy protection.** The DP does not need to provide a real ID for data transaction validation, that protects the trading status of the DP. Meanwhile, zero-knowledge arguments will not disclose the information about the DP and data keys, only *DL* access permission transfer can be observed on the public channel. Therefore, the proposed solution can achieve the protection of transaction and identity privacy information of data providers. Consumers who have purchased data, can obtain the DP's identity information through the association of trading transactions.

6.2 Formal Security Verification

We use the formal verification tool Tamarin [37] to perform formal security verification of the proposed authentication and key distribution protocol. Tamarin prover is an effective tool, which has been applied to numerous protocols including Transport Layer Security (TLS) [38], for symbol modeling and analysis of security protocols.

After inputting the security protocol model to Tamarin, it will specify the agents to perform the behavior simulations of different roles, the rules for the adversary and the rules for the security properties required by the protocol. And then Tamarin will automatically construct a security proof. We rewrite "rules" to define protocols and adversaries using an expressive language based on multiset. Subsequently, we convert the protocol into executable code in the Tamarin tool by the following four steps. (1) Function construction, where the equations and cryptographic algorithms within the protocol are written as functions to be called by the protocol. Some of the cryptographic algorithms such as hashing and asymmetric encryption are already included in the Tamarin tool. (2) The infrastructure (e.g., authentication server) is modeled and the protocol environment is described using multiset rewriting rules that operate on the state of the system. (3) The proposed protocol is modeled by invoking the above rules and functions to compose a protocol flow that simulates the message transfer and computational operations between entities. (4) The security properties are modeled. In our protocol, we need to verify the mutual authentication between the data provider and the data consumers, the confidentiality and the integrity of keys, the resistance of the protocol to attacks, and other security properties. In this simulation, we write 6 types of rules to represent the actions of the proposed protocol, which include initialization, nonce generation, registration, key revealation, message transmission, and message validation, as shown in Fig. 7.

For the authentication properties of the protocol, Tamarin formalizes the entity resistance to attacks, such as *Aliveness* for fake identity, *Weak agreement* for replay attacks and *Injective agreement* for Man-in-the-Middle attacks. Other claims like *Executable* and *Secret* represent the enforceability and confidentiality of the protocol, respectively. We model the data provider and multiple data consumers in Tamarin. As shown in Fig. 8, the simulation results indicate that the multi-party entities in our protocol can be authenticated securely and are resistant to various attacks, with no valid attacks found under the Tamarin simulation.

```
rule platform_init:
  let
      pubkey = pk(~prikey)
  in
    [ Fr(~prikey) ]
  -->
    [ !Platform($A, ~prikey, pubkey), Out(pubkey) ]
```

Initialization Rule

```
rule Platform_generate_Rand_nounce:
    [ Fr(~RN_Platform), !Platform($A, ~prikey, pubkey)]
  --[ Platform_generate_RN($A, ~RN_Platform) ]->
    [ !RN_generate_by_Platform($A, ~RN_Platform) , Out(pubkey) ]
```

Nounce Generation Rule

```
rule register_identities:
  let
      pubkey = pk(~prikey)
  in
    [ Fr(~prikey), Fr(~ID), Fr(~s) ]
  -->
    [ Node($A, ~ID, ~s, ~prikey, pubkey), Out(pubkey), Out(~ID) ]
```

Registration Rule

```
rule Reveal_Key:
[DP($DP, ID_DP, s_DP, SK_DP, PK_DP, GS) ]--[Reveal(<'DP',GS>)]->[Out(GS)]
```

Key Revealation Rule

```
rule DC1_send_M_2:
  let
     pi = M_1
     M_2_DC1 = <sign(<h(pi), ~T_2_DC1>, SK_DC1), ID_DC1, ~T_2_DC1>
  in
    [ Fr(~T_2_DC1), Node($DC1, ID_DC1, s_DC1, SK_DC1, PK_DC1),
      In(<DP, $DC1, M_1, h(s_DP), RN_Platform, T_1, MI, PK_DP>) ]
  --[ Eq(verify(pi, <h(s_DP), RN_Platform, T_1>, PK_DP), true),
      Recv($DC1, <M_1, h(s_DP), RN_Platform, T_1>), Send($DC1, M_2_DC1),
      Authentic(DP, M_1), Running(DP,$DC1,<'DC1','DP',M_1, M_2_DC1>)
       ]->
    [ Out(<$DC1, DP, M_2_DC1, PK_DC1, ID_DC1, ~T_2_DC1>),
      St_DC1_1($DC1, ID_DC1, s_DC1, SK_DC1, PK_DC1, h(s_DP), RN_Platform, ~T_2_DC1)]
```

Message Transmission Rule

```
rule DP_verify_M_2_DC1_send_M_3:
  let
     M_3_DC1 = aenc(<GS, ~RN_3, T_2_DC1, ~T_3_DC1, ID_DC1>, PK_DC1)
  in
    [ Fr(~RN_3), Fr(~T_3_DC1),
      In(<$DP, $DC1, h(pi), M_2_DC1, ID_DC1, T_2_DC1, PK_DC1>),
      DP_1($DP, ID_DP, s_DP, SK_DP, PK_DP, GS, T_1, RN_Platform)
    ]
  --[ Eq(verify(fst(M_2_DC1), <h(pi), T_2_DC1>, PK_DC1), true),
      Send($DP, M_3_DC1), Authentic($DP, M_3_DC1),Running($DC1,$DP,<'DC1','DP',M_2_DC1>),
      Commit_M_3($DP, $DC1, M_3_DC1) ]->
    [ Out(<M_3_DC1, ~T_3_DC1>),
      DP_2($DP, ID_DP, s_DP, SK_DP, PK_DP, GS, T_1, RN_Platform, ~T_3_DC1, h(pi), M_2_DC1, T_2_DC1, PK_DC1) ]
```

Message Validation Rule

Figure 7: Protocol modeling in Tamarin

```
Terminal
File  Edit  View  Search  Terminal  Help

executable (exists-trace): verified (15 steps)
M_1_DP_authentication (exists-trace): verified (12 steps)
alivenessDP_DC1 (all-traces): verified (3 steps)
alivenessDP_DC2 (all-traces): verified (3 steps)
alivenessDC1_DP (all-traces): verified (3 steps)
alivenessDC2_DP (all-traces): verified (3 steps)
weak_agreementDP_DC1 (all-traces): verified (10 steps)
weak_agreementDP_DC2 (all-traces): verified (10 steps)
weak_agreementDC1_DP (all-traces): verified (10 steps)
weak_agreementDC2_DP (all-traces): verified (10 steps)
noninjective_agreementDP_DC1 (all-traces): verified (10 steps)
noninjective_agreementDP_DC2 (all-traces): verified (10 steps)
noninjective_agreementDC1_DP (all-traces): verified (4 steps)
noninjective_agreementDC2_DP (all-traces): verified (4 steps)
injectiveagreementDP_DC1 (all-traces): verified (10 steps)
injectiveagreementDP_DC2 (all-traces): verified (10 steps)
injectiveagreementDC1_DP (all-traces): verified (4 steps)
injectiveagreementDC2_DP (all-traces): verified (4 steps)
GS_is_secret (all-traces): verified (3 steps)

==============================================================================
```

Figure 8: Protocol security simulation results in Tamarin

7 Performance Evaluation

Performance includes the following two parts: the proposed protocol performance and blockchain service performance. On the protocol performance, we use algorithm simulation to evaluate it by comparing with other solutions in terms of computational cost and transmission cost. On blockchain service performance, we use the Hyperledger Caliper framework to evaluate the performance of Fabric in terms of permission distribution throughput and transaction confimration delay.

This work is based on the Hyperledger Fabric version 1.4.0. Firstly, we deploy the Golang environment, Docker, and Docker Compose. The system uses Docker to establish a virtual machine for multi nodes environment simulation. The zero-knowledge proof adopted in this scheme is developed based on the Libsnake framework. For the convenience of use, we compile the executable binary ZKP. The tools and versions used during the compilation process are as shown in Table 3.

Table 3: Zero-knowledge proof compiler version

Tools	versions
gcc	gcc (GCC) 4.8.5 20150623 (Red Hat 4.5.5-44)
golang	go version gol.16 linux/amd 64
stack	The haskell Tool Stack, Version 2.7.1

For the web service environment of trading platform, the testing browser is Microsoft Edge, and the specific version information is as shown in Table 4.

Table 4: Web service environment

Contents	versions
Microsoft edge	91.0.864.37 (64-bit)
Operating system	Windows 10 OS
JavaScript	V8 9.0.269.28
Client agent	Mozilla/5.0 (Windows NT 10.0, Win 64, x64)

7.1 Proposed Protocol Performance

7.1.1 Computational Cost

In this section, we simulate and calculate the time consumed by the proposed protocol by comparing the results with other related schemes in [24,25,28,30]. We use the MIRACL library algorithms and the libsnake library for zero-knowledge proof to evaluate time consumption. The symmetric and asymmetric encryption algorithm adopts AES-256 and ECC-164, respectively. The results are shown in Table 5.

For **n** data consumers, each purchasing **m** pieces of data, the time consumption comparison of the solutions is shown in Table 6.

As shown in Fig. 9, the proposed protocol has the advantage on the computational efficiency when the number of data consumers or the number of datasets increases, because our solution can aggregate and verify data consumers, as well as simultaneously verify the ownership of multiple sets of

data. Specifically, in our protocol, data consumers do not need to perform more validation calculations when purchasing more datasets.

Table 5: Cryptographic algorithm time cost

Notations	Definition	Time cost (ms)
T_{E1}	Exponentiation operation in $Z^*_{N^2}$	11.794
T_{E2}	Exponentiation operation in $\mathbb{G}$	11.926
T_M	Multiplication operation in $\mathbb{G}$	0.442
T_S	Algebraic sum operation in $\mathbb{G}$	0.0018
T_{Pro}	Prove operation for NIZK	3.831
T_{Vry}	Verify operation for NIZK	1.594
T_P	Pairing operation	28.63
T_{EE}	Encryption operation for ECC	3.274
T_{ED}	Decryption operation for ECC	3.524

Table 6: Time cost comparison

Schemes	DP	DC
Ours	$(3n+1)T_M + (4n+1)T_S + T_{Pro} + nT_{EE}$	$3T_m + T_S + T_{Vry} + T_{ED}$
[24]	$2mn(T_{ED})$	$3mT_{EE}$
[25]	$mn(4T_{E2} + 4T_{E1} + 6T_M) + mT_P$	$m(T_M + 3T_{E2} + T_P)$
[28]	$m(4n^2 + 2n + 1)T_M + m(n^2 + 2n + 2)T_{E1}$	$3mT_M + 2mT_{E1}$
[30]	$nm(T_{Pro} + T_{Vry} + 2T_s)$	$m(T_{Pro} + T_{Vry})$

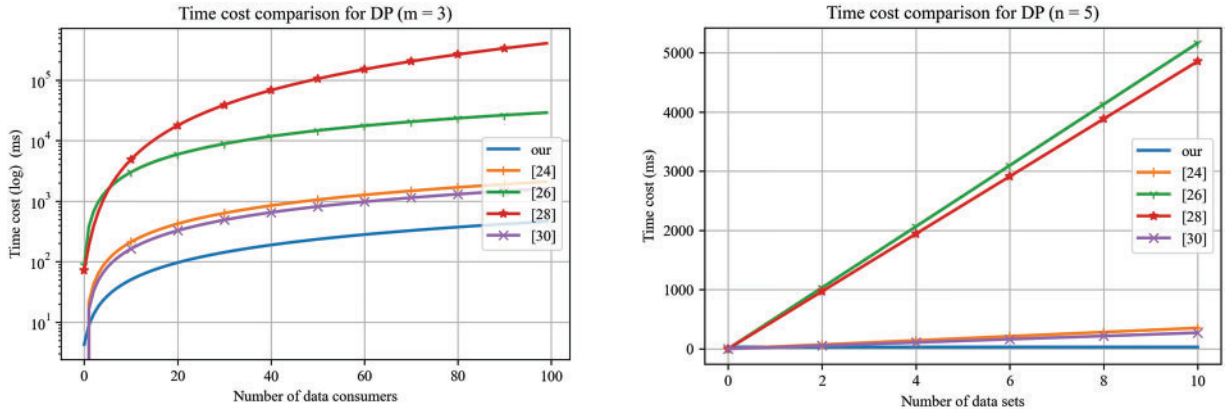

Figure 9: (Continued)

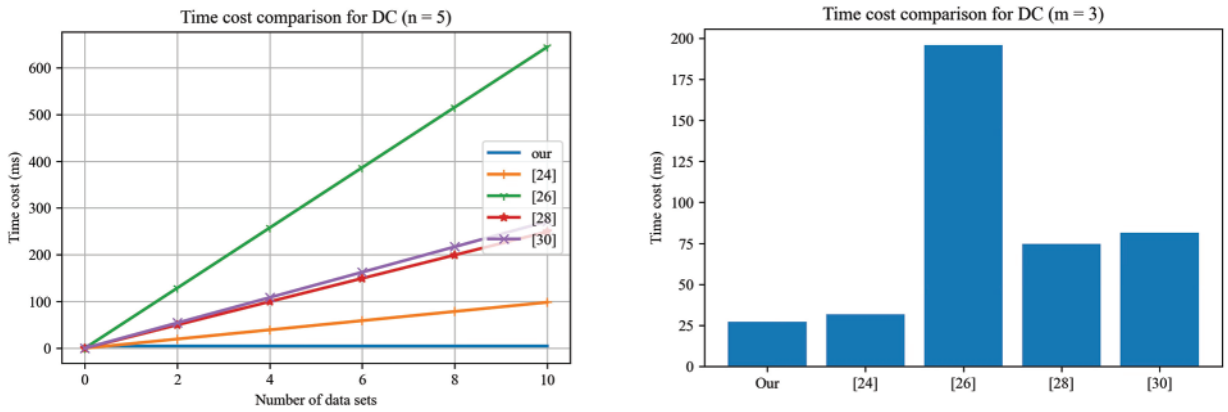

Figure 9: Time costs comparison

7.1.2 Transmission Overhead

On the transmission overhead, the size of transmission messages are evaluated other than the trading data itself. In platform construction, the hash output length of the elliptical curve is 512 bits and the *ID*, timestamp is 16 bits. And *Proof* size for Groth 16 in our scheme is 0.2 kb which is 0.08% of the proof size in Factual (250 kb) [13]. The transmission overhead comparison is shown in Table 7 and Fig. 10. As shown in Fig. 10, the proposed protocol has fewer interaction rounds and lower message transmission consumption for verification than other related schemes. Similarly, when purchasing multiple data sets, data consumers do not need to incur additional transmission costs.

Table 7: Transmission overhead

Schemes	DP	DC
Ours	$32m + 16mn + 288n + 1054$ bits	$1056 + 16m$ bits
[24]	$180 + 544mn$ bits	$180 + 1728m$ bits
[25]	$1600mn$ bits	$1072m$ bits

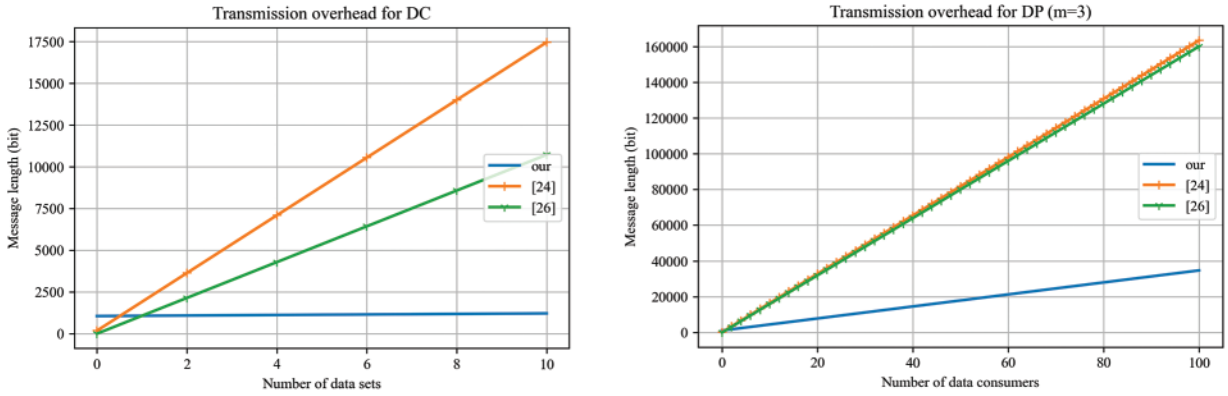

Figure 10: Transmission Overhead Comparison

7.2 Blockchain Service Performance

Caliper is a benchmark testing framework specifically designed to test the permission distribution throughput and transaction confirmation delay. Currently, it supports multiple performance indicators, including transaction success rate, transaction throughput, and transaction latency. We simulate the impact of different transaction request rates on Fabric throughput and transaction latency when the block size is 50 MB. The following data are obtained as shown in Table 8.

Table 8: Permission distribution throughput and transaction confirmation delay

Request rates (tps)	20	40	60	80	100	120	140	160	180
Throughput (tps)	18	39	58	74	73	75	74	76	77
Latency (s)	0.7	0.5	0.8	0.9	1.3	1.6	2.6	2.1	2.2

As shown in Fig. 11, as the transaction request rate increases, the initial throughput also increases linearly, but after reaching a certain bottleneck (around 80 tps), the throughput maintains about 75 tps. When the transaction request rate is low, the transaction latency is maintained at around 0.6 s, but when the transaction request rate is higher than around 80 tps, the delay of the transaction starts to increase. Overall, the transaction confirmation time on this platform is in seconds, which is user-friendly.

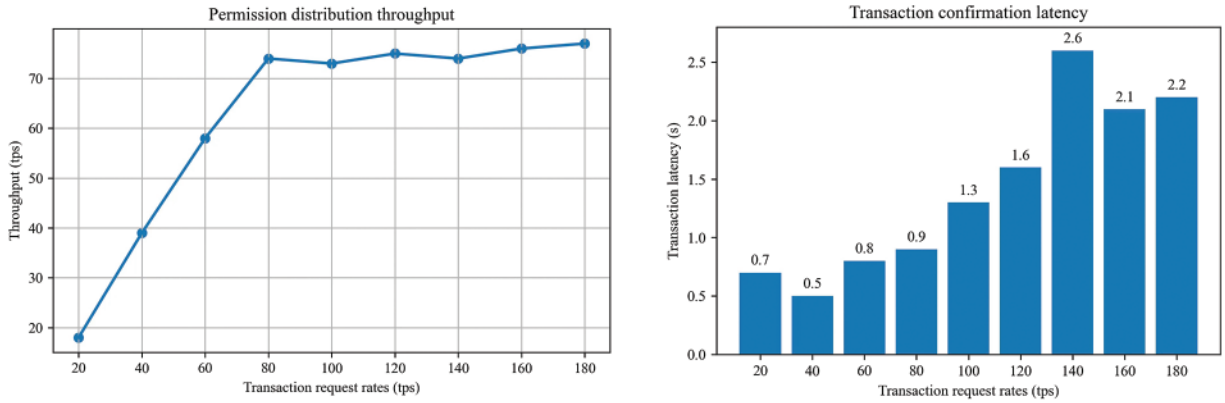

Figure 11: Permission distribution throughput and transaction confirmation delay

8 Conclusion

More and more intelligent sensing devices are connected to the internet, causing a rapid growth of networking data volume. The collaborative, integrated, and intelligent development of the Internet of Things requires data sharing and transaction support between multiple domain nodes. We designed a decentralized IoT secure data sharing platform that allows individual users to freely and conveniently join. The fog and cloud enabled trading platform realizes decentralized identity management and trading organization, balancing the security and the efficiency of data sharing. For the identity authentication module of the platform, we also innovatively combine the zero-knowledge proof technology with aggregated signature, authenticate the data provider and data ownership through the zero-knowledge proof identity authentication algorithm. The supplier simultaneously verifies the identities of multiple consumers through the aggregated signature algorithm. The proposed scheme can

realize the secure one to many data sharing under the protection of transaction sensitive information, and reduce the computational overhead and transmission overhead, which is very suitable for the large amount of secure data sharing among resource-limited devices.

Acknowledgement: The authors would like to acknowledge Ministry of Science and Technology of the People's Republic of China and National Natural Science Foundation of China (NSFC) for the financial support through National Key R&D Program of China (No. 2022YFB3103400) and NSFC under Grant 61932015 and Grant 62172317. The authors would like to give thanks to the School of Cyber Engineering, Xidian University, China, for promoting this research and providing the laboratory facility and support.

Funding Statement: This work is supported by the National Key R&D Program of China (No. 2022YFB3103400), the National Natural Science Foundation of China under Grants 61932015 and 62172317.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Y. Luo, J. Cao; methodology: Y. Luo, W. You; data collection: C. Sahng, Y. Luo; analysis and interpretation of results: Y. Luo, C. Shang; draft manuscript preparation: X. Ren, Y. Luo, C. Shang; supervision: H. Li. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Li, S., Xu, L., Zhao, S. (2018). 5G Internet of Things: A survey. *Journal of Industrial Information Integration, 10,* 1–9.
2. Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M. (2018). Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials, 20(4),* 2923–2960.
3. Aklilu, Y. T., Ding, J. (2022). Energy big data security threats in IoT-based smart grid communications. *Energies, 15(193),* 1–26.
4. Kumar, P., Lin, Y., Bai, G., Paverd, A., Dong, J. S. et al. (2019). Smart grid metering networks: A survey on security, privacy and open research issues. *IEEE Communications Surveys & Tutorials, 21(3),* 2886–2927.
5. Chin, W. L., Li, W., Chen, H. H. (2017). Survey on blockchain for smart grid management, control, and operation. *IEEE Communications Magazine, 55(10),* 70–75.
6. Gopstein, A., Hastings, N., Feldman, L., Agarwal, R., Bartol, N. (2021). Distributed energy resource security: Potential guidelines and research topics. *Technical Note (NIST TN)*, National Institute of Standards and Technology.
7. Butt, O. M., Zulqarnain, M., Butt, T. M. (2021). Recent advancement in smart grid technology: Future prospects in the electrical power network. *Ain Shams Engineering Journal, 12(1),* 687–695.
8. Mollah, M. B., Zhao, J., Niyato, D., Lam, K. Y., Zhang, X. et al. (2021). Blockchain for future smart grid: A comprehensive survey. *IEEE Internet of Things Journal, 8(1),* 18–43.
9. Laroui, M., Nour, B., Moungla, H., Cherif, M. A., Afifi, H. et al. (2021). Edge and fog computing for IoT: A survey on current research activities & future directions. *Computer Communications, 180,* 210–231.
10. Zhuang, P., Zamir, T., Liang, H. (2020). Blockchain for cybersecurity in smart grid: A comprehensive survey. *IEEE Transactions on Industrial Informatics, 17(1),* 3–19.

11. Pillitteri, V., Brewer, T. (2014). Guidelines for smart grid cybersecurity. *NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology*. https://doi.org/10.6028/NIST.IR.7628r1
12. Voigt, P., von dem Bussche, A. (2017). The EU general data protection regulation (GDPR). In: *A practical guide*, vol. 2017, 1st edition. Cham: Springer International Publishing.
13. Elbaz, G. (2023). Factual Data. https://www.factualdata.com/ (accessed on 03/04/2023).
14. Wang, Y., Su, Z., Zhang, N., Chen, J., Sun, X. et al. (2021). SPDS: A secure and auditable private data sharing scheme for smart grid based on blockchain. *IEEE Transactions on Industrial Informatics, 17(11),* 7688–7699.
15. Lu, X., Fu, S., Jiang, C., Lio, P. (2021). A fine-grained iot data access control scheme combining attribute-based encryption and blockchain. *Security and Communication Networks, 2021,* 1–13.
16. Nasiraee, H., Ashouri-Talouki, M. (2022). Privacy-preserving distributed data access control for CloudIoT. *IEEE Transactions on Dependable and Secure Computing, 19(4),* 2476–2487.
17. Zhang, P., Chen, Z., Liu, J. K., Liang, K., Liu, H. (2018). An efficient access control scheme with outsourcing capability and attribute update for fog computing. *Future Generation Computer Systems, 78,* 753–762.
18. Feng, C., Yu, M., Aloqaily, M., Alazab, M., Lv, Z. et al. (2020). Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV. *IEEE Transactions on Vehicular Technology, 69(1),* 13784–13795.
19. Sun, Y., Lin, L., Sun, Z., Tian, Z., Du, X. (2020). An IoT data sharing privacy preserving scheme. *Proceedings of IEEE Conference on Computer Communications Workshops (INFOCOMWKSHPS)*, pp. 984–990.
20. Tan, L., Shi, N., Aloqaily, M., AJararweh, Y. (2021). A blockchain-empowered access control framework for smart devices in green Internet of Things. *ACM Transactions on Internet Technology, 21(3),* 1–20.
21. Sheng, D., Xiao, M., Liu, A., Zou, X., An, B. et al. (2020). CPchain: A copyright-preserving crowdsourcing data trading framework based on blockchain. *Proceedings of 2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–9.
22. Cheng, G., Deng, S., Xiang, Y., Chen, Y., Yin, J. (2020). An auction-based incentive mechanism with blockchain for IoT collaboration. *Proceedings of 2020 IEEE International Conference on Web Services (ICWS)*, pp. 17–26. Beijing, China.
23. Li, H., Pei, L., Liao, D., Wang, X., Xu, D. et al. (2021). BDDT: Use blockchain to facilitate IoT data transactions. *Cluster Computing, 24(1),* 459–473.
24. Dixit, A., Singh, A., Rahulamathavan, Y., Rajarajan, M. (2023). Fast data: A fair, secure and trusted decentralized IIoT data marketplace enabled by blockchain. *IEEE Internet of Things Journal, 10(4),* 2934–2944.
25. Yu, K., Tan, L., Aloqaily, M., Yang, H., Jararweh, Y. (2021). Blockchain enhanced data sharing with traceable and direct revocation in IIot. *IEEE Transactions on Industrial Informatics, 17(11),* 7669–7678.
26. Boo, E., Kim, J., Ko, J. (2022). LiteZKP: Lightening zeroknowledge proof-based blockchains for IoT and edge platforms. *IEEE Systems Journal, 16(1),* 112–123.
27. Özyilmaz, K. R., Dogan, M., Yurdakul, A. (2018). IDMOB: IoT data marketplace on blockchain. *Proceedings of 2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 11–19. Theater Casino Zug, Switzerland.
28. Chen, S., Yang, L., Zhao, C., Varadarajan, V., Wang, K. (2022). Double-blockchain assisted secure and anonymous data aggregation for fog-enabled smart grid. *Engineering, 8,* 159–169.
29. Zhang, Y., He, D., Choo, K. K. R. (2018). BADS: Blockchain-based architecture for data sharing with ABS and CP-ABE in IoT. *Wireless Communications and Mobile Computing, 2018,* 1–9.
30. Walshe, M., Epiphaniou, G., Al-Khateeb, H., Hammoudeh, M., Katos, V. et al. (2019). Non-interactive zero knowledge proofs for the authentication of iot devices in reduced connectivity environments. *Ad Hoc Networks, 95,* 101988.1–101988.12.

31. Groth, J. (2016). On the size of pairing-based non-interactive arguments. *Proceedings of Advances in Cryptology EUROCRYPT 2016*, pp. 305–326. Vienna, Austria.
32. Koblitz, N., Menezes, A., Vanstone, S. (2000). The state of elliptic curve cryptography. *Designs, Codes and Cryptography, 19,* 173–193.
33. Ding, H., Chen, X., Lin, D. (2021). IEEE standard for framework of blockchain-based Internet of Things (IoT) data management. In: *IEEE Std 2144.1-2020*, pp. 1–20.
34. Dolev, D., Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory, 29(2),* 198–208.
35. Chen, Y., Li, M., Zhu, X., Fang, K., Ren, Q. et al. (2022). An improved algorithm for practical byzantine fault tolerance to large-scale consortium chain. *Information Processing & Management, 59(2),* 102884.
36. Zhan, Y., Wang, B., Lu, R. (2020). Cryptanalysis and improvement of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks. *IEEE Internet of Things Journal, 8(8),* 5973–5984.
37. Meier, S., Schmidt, B., Cremers, C., Basin, D. (2013). The tamarin prover for the symbolic analysis of security protocols. *Proceedings of International Conference on Computer Aided Verification*, pp. 696–701. St. Petersburg, Russia.
38. Cremers, C., Horvat, M., Scott, S., van der Merwe, T., (2016). Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication. *Proceedings of 2016 IEEE Symposium on Security and Privacy (SP)*, pp. 470–485. San Jose, Canda.