Lightweight and Secure Data Sharing Based on Proxy Re-Encryption for Blockchain-Enabled Industrial Internet of Things

Fengqun Wang, Jie Cui, *Senior Member, IEEE*, Qingyang Zhang, *Member, IEEE*, Debiao He, *Member, IEEE*, Chengjie Gu, and Hong Zhong, *Member, IEEE*

***Abstract*—In the Industrial Internet of Things (IIoT), data sharing is crucial for promoting the intelligent development of industrial production. To achieve effective data supervision, introducing blockchain into traditional cloud-based data-sharing frameworks has attracted widespread attention. However, existing blockchain-based data-sharing schemes still have issues with security and efficiency. Therefore, we propose a blockchain-enabled data-sharing scheme based on proxy re-encryption. First, the scheme considers both storage and access authentication, guaranteeing data sources' trustworthiness and preventing data misuse. Second, the scheme uses an on-chain and off-chain cooperative storage mechanism, saving the storage resources of the blockchain. Third, the scheme supports data packing, which effectively improves data storage efficiency. The security analysis shows that our scheme satisfies the security requirements. Finally, we build a blockchain platform using the hyperledger fabric. The performance evaluation shows that our scheme is more advantageous regarding computational overhead than other related schemes.**

***Index Terms*—Authentication, blockchain, data sharing, elliptic curve cryptography (ECC), Industrial Internet of Things (IIoT), proxy re-encryption.**

I. Introduction

THE INDUSTRIAL Internet of Things (IIoT) [1] has integrated various smart devices (SDs) and emerging technologies into industrial production, improving production efficiency and reducing production costs [2], [3]. To optimize the production model and enhance decision-making efficiency [4], [5], [6], production data (e.g., temperature) from SDs are often shared. For example, when a batch of products has quality problems, the product analyst needs to analyze the production data and optimize the production strategy. However, data sharing brings security and efficiency challenges to the IIoT. On the one hand, for privacy-sensitive industrial departments, exposing large amounts of private data to an open network is intolerable [7], [8] because it may cause industrial decision errors. On the other hand, SDs are resource limited [9]; they cannot afford the computational and storage overheads associated with processing massive amounts of industrial data.

To guarantee the security and efficiency of data sharing, researchers have paid extensive attention to proxy re-encryption [10], [11] and cloud-computing technology [12], [13]. Fig. 1 illustrates a general cloud-based data-sharing framework with proxy re-encryption in the IIoT. For example, to optimize production decisions, a product analyst [as a data user (DU)] can access industrial data generated by SDs through the framework. Specifically, the edge server (ES) periodically collects the encrypted data generated by the SD and subsequently uploads it to the cloud server (CS). Once the cloud service receives an access request from a product analyst, it performs re-encryption operation to transform the ciphertext into ciphertext that can be decrypted

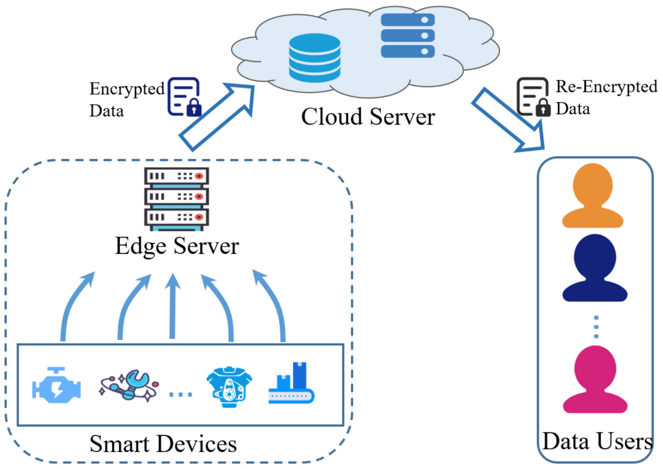


Fig. 1. Cloud-based data-sharing framework in IIoT.

Manuscript received 3 October 2023; revised 25 November 2023; accepted 2 December 2023. Date of publication 8 December 2023; date of current version 9 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62272002, Grant U23A20308, and Grant 62202005; in part by the Excellent Youth Foundation of Anhui Scientific Committee under Grant 2108085J31; in part by the Natural Science Foundation of Anhui Province, China, under Grant 2208085QF198; in part by the Open Fund of Anhui Province Key Laboratory of Cyberspace Security Situation Awareness and Evaluation under Grant CSSAE-2021-008; and in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2022-049. *(Corresponding author: Jie Cui.)*

Fengqun Wang, Jie Cui, Qingyang Zhang, and Hong Zhong are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, the Anhui Engineering Laboratory of IoT Security Technologies, and the Institute of Physical Science and Information Technology, Anhui University, Hefei 230039, China (e-mail: cuijie@mail.ustc.edu.cn).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: hedebiao@163.com).

Chengjie Gu is with the Security Research Institute, New H3C Group, Hefei 230088, China (e-mail: gu.chengjie@h3c.com).

Digital Object Identifier 10.1109/JIOT.2023.3340567

by the product analyst. By utilizing proxy re-encryption, this framework achieves data sharing without compromising the privacy of the original data. However, the framework relies entirely on third-party CSs and lacks effective supervision of data storage, access, and processing [14], [15]. For example, to maximize profits, cloud service providers may return incorrect ciphertext to DUs, leading to incorrect production decisions.

To solve these problems, the blockchain technology is considered a promising solution owing to its decentralized, tamper-proof, and traceability features [16], [17], [18]. Although some blockchain-based data-sharing schemes have been proposed, the deficiencies in terms of efficiency and security still need further improvement.

1) In the IIoT system, different entities subscribe to services based on production tasks. Only the entities that have subscribed to the corresponding services can store and access the data. Therefore, the blockchain must perform storage/access authentication to determine whether the data uploader/user has subscribed to the corresponding service. However, most existing schemes fail to consider this requirement fully. For example, in scheme [19], the blockchain only verifies the identity of the data uploader and does not further determine whether the data uploader is subscribed to the corresponding service. In this case, the blockchain may store illegitimate data. In scheme [20], the blockchain does not determine whether the DU is subscribed to the corresponding service. In this case, shared data can be misused, and IIoT privacy can be compromised. Therefore, we must design secure storage and access authentication algorithms.
2) The industrial data generated by SDs is massive and needs to be processed quickly. However, in some schemes, the blockchain needs to process a large number of requests and store large-scale data, resulting in low performance. For example, in scheme [21], SDs submit data storage requests directly to the blockchain and store encrypted data in the blockchain. On the one hand, there are too many requests submitted by SDs while the throughput of the blockchain is limited [22], [23], so the blockchain cannot respond in time. On the other hand, the blockchain needs to store too much data while its storage performance is low [24], so it generates significant time delays. Therefore, we urgently need a mechanism to reduce the processing and storage pressure on the blockchain.

Inspired by the scheme [25], we design a secure and lightweight data-sharing scheme to solve the above issues. Moreover, the proposed scheme not only achieves effective supervision of the data but also exhibits low computational overhead and high security. The contributions of the proposed scheme are as follows.

1) We design a data sharing scheme based on proxy re-encryption, effectively ensuring secure data sharing. To ensure that the data source is trusted and to prevent data misuse, the proposed scheme provides storage and access authentication. Data can only be shared between ESs and DUs that subscribe to the same service.
2) The scheme realizes on-chain supervision and off-chain storage, effectively reducing the storage pressure on the blockchain. Second, the framework uses an ES to package and upload data from SDs, effectively avoiding direct interaction between SDs and the blockchain, which reduces the number of requests processed by the blockchain.
3) The security analysis shows that our proposed scheme is highly secure. The performance analysis results show that our proposed scheme is effective and practical for blockchain-enabled data-sharing IIoT.

The remainder of this article is organized as follows. Section II describes existing relevant data-sharing schemes and analyzes their limitations and applicability in the IIoT. Section III presents preliminaries. Section IV details the blockchain-based data-sharing framework. Section V details our scheme. Section VI shows the security proof and analysis. Section VII presents the performance evaluation. Finally, Section VIII summarizes the study.

II. Related Work

In recent years, to achieve secure and efficient data sharing, a series of research works have been proposed.

Attribute-based encryption [26], [27] is often used in data-sharing schemes because it enables fine-grained access control while ensuring data confidentiality. For example, Li et al. [28] considered that outsourced decryption does not guarantee the correctness of the transformations performed by the CS and proposed an attribute-based encryption scheme. The scheme can check the correctness of the converted ciphertext for authorized and unauthorized users. Ning et al. [29] proposed a cloud-based data storage and sharing scheme. The scheme utilizes attribute encryption technology to achieve access authentication for DUs. Guo et al. [30] proposed a revocable blockchain-assisted attribute encryption scheme. The scheme allows CSs to perform predecryption operations to reduce the computational overhead of DUs. To solve the problems of user key abuse and authorization center key abuse, Li et al. [31] proposed an attribute-based encryption access control system and constructed an accountable data-sharing scheme.

Proxy re-encryption techniques [32] are often used in data-sharing schemes because they enable ciphertext transformation without compromising data privacy. Specifically, this technique allows agents to transform the original ciphertext into a ciphertext that can be decrypted by other legitimate DUs, thus achieving flexible data sharing. Xu et al. [33] proposed a certificateless proxy re-encryption scheme, which achieves data access control without fully trusting the CS. In this scheme, the agent cannot perform the re-encryption operation without obtaining the re-encryption key generated by the data owner, avoiding the abuse of data sharing. For the security and trustworthiness of the data, Ge et al. [34] presented a verifiable and fair attribute-based proxy re-encryption scheme. In this scheme, the DU can determine whether the data returned by the CS is correct, ensuring that the CS is not maliciously accused. Su et al. [35] proposed a cloudIoT platform based on proxy re-encryption, which achieves secure data sharing. In addition, this platform introduces an uploading authorization server and a download authorization server, implementing storage and access authentication.

Although the above studies ensure data security in different ways, they contain many time-consuming cryptographic operations, such as pairing, which will impose a heavy computational burden on resource-limited IIoT devices. In addition, they do not fully realize the supervision of data storage, access, and processing.

Considering the limitations of data supervision, some researchers have started introducing blockchain into the Internet of Things (IoT) [25], [36]. Although these schemes do not give a specific data-sharing algorithm, the idea of on-chain supervision and off-chain storage mentioned is instructive.

Agyekum et al. [37] proposed a data-sharing scheme using blockchain. The scheme combines proxy re-encryption and identity-based encryption to ensure secure data sharing. However, the scheme exposes the real identity of the data owner, which is not conducive to protecting the privacy of SDs in IIoT systems. Manzoor et al. [20] proposed a blockchain-based data-sharing platform. The platform uses elliptic curve cryptography (ECC) to design a lightweight proxy re-encryption algorithm that enables efficient and secure data sharing for resource-constrained IoT environments. Unfortunately, the scheme does not give a specific storage and access authentication algorithm. Chen et al. [21] proposed a lightweight proxy re-encryption scheme. The scheme utilizes the blockchain technology and equality testing techniques to ensure that data is shared securely. However, in this scheme, SDs store data in the blockchain directly, which increases the storage burden of the blockchain and is only suitable for IIoT environments with low data volumes. In addition, the scheme pays no attention to protecting the anonymity of SDs. Lu et al. [19] proposed a blockchain-based data storage and sharing scheme. The scheme combines group signature and proxy re-encryption technology to provide storage and access authentication, effectively ensuring the anonymity of SDs and the trustworthiness of data sources. However, the group signature contains many time-consuming operations that impose a huge computational burden on the blockchain-based data-sharing platform.

The above analysis shows that the current data-sharing schemes still have some limitations and deficiencies when used in IIoT environments. Therefore, it is meaningful and necessary to design a secure and lightweight data-sharing scheme that can supervise the data effectively.

III. Preliminaries

In this section, we introduce preliminaries from two aspects: 1) elliptic curve cryptosystem and 2) hash function.

A. Elliptic Curve Cryptosystem

Let $\mathbb{F}_p$ be a finite field and $E : by^2 = x^3 + ax^2 + x$ be a nonsingular elliptic curve over $\mathbb{F}_p$, where $a, b \in \mathbb{F}_p$ and $b(a^2 - 4) \bmod q \neq 0$. Let $\mathbb{G}$ be a cyclic group on E of prime order q.

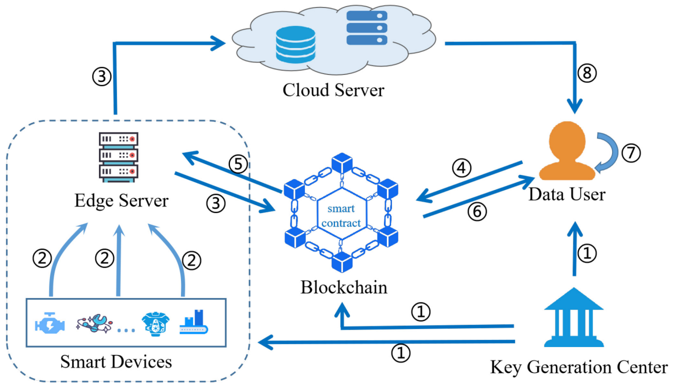


Fig. 2. Blockchain-based data-sharing framework in IIoT.

1) *Definition 1: (Discrete Logarithm (DL) Problem):* Let $P, Q \in \mathbb{G}$, $x \in \mathbb{Z}_q^*$, and $Q = x \cdot P$. Given P and Q, it is hard to obtain x in a probabilistic polynomial time (PPT).
2) *Definition 2: (Computational Diffie–Hellman (CDH) Problem):* Let $x, y \in \mathbb{Z}_q^*$ and given three points $P, xP, yP \in G$, it is hard to get xyP in a PPT.

B. Hash Function

The hash function can compress any length of data into a shorter fixed length of data. We use secure hash functions with the following properties.

1) *One Way:* Given x as the input of a hash function $H(\cdot)$, it can obtain y by computing $y = H(x)$. However, given y, it is hard to obtain x by calculating $x = h^{-1}(y)$.
2) *Collision Resistant:* Given x, it is computationally impossible to find another data $x' \neq x$ such that $H(x') = H(x)$.

IV. Background

This section presents the blockchain-based data-sharing framework, threat model, and design goals.

A. Blockchain-Based Data-Sharing Framework

We consider the scenario of data sharing in IIoT. In the case of a smart factory, the ES regularly collects the production data of SDs and stores it in the CS. Once the products have quality issues, product analysts, as DUs, can access this production data.

1) Entity Overview: As shown in Fig. 2, the framework contains six entities: 1) key generation center (KGC); 2) ES; 3) DU; 4) SDs; 5) blockchain (BC); and 6) CS.

1) *KGC* is trusted by other entities in the IIoT. It is mainly responsible for generating system parameters.
2) *ES* is responsible for collecting, sorting, packaging, and sharing data from SDs.
3) *DU* is mainly responsible for accessing the shared data and restoring the shared data to the original data. It can access data based on its subscribed services.
4) *SD* has limited computing power and storage capacity. It is the generator and publisher of the original data.

5) *CS* is a third-party server with strong storage capacity. It is primarily responsible for storing the packaged data processed by ES.
6) *BC* is a trusted platform in IIoT that can run smart contracts, mainly responsible for storing metadata (hides the assisted secret key corresponding to original data and the index of packaged data), performing storage, and access authentication.

2) High-Level Workflow: To achieve data sharing, the high-level workflow consists of the following eight steps.

1) KGC initializes the system and sends the parameters to the corresponding entities.
2) SD encrypts and signs the original data. Then, the SD sends the encrypted data to ES.
3) ES collects and packages data from SDs, stores the packaged data in CS, and stores the corresponding metadata in BC.
4) DU generates an access request and sends it to BC.
5) BC verifies the access request from DU, and then requests a re-encryption key from ES.
6) BC generates a re-encryption metadata and sends it to the DU.
7) DU obtains the assisted secret key and packaged data index by decrypting the re-encryption metadata.
8) DU utilizes the index to retrieve the corresponding packaged data from CS, and then the DU decrypts the packaged data to obtain the original data.

B. Threat Model

In our proposed scheme, the attackers mainly come from outside the system; they can launch both active and passive attacks. Specifically, when attackers launch active attacks, they mainly tamper with data in the IIoT system; when they launch passive attacks, they mainly listen to data in the IIoT and try to mine private information from it.

C. Design Goals

1) Security Goals: Our proposed scheme aims to achieve the following security goals.

1) *Confidentiality:* The original data contains critical information for industrial production, so the confidentiality of the original data needs to be guaranteed. First, the network attacker cannot obtain the original data. Second, the network attacker cannot get the assisted secret key and packaged data index.
2) *Integrity:* The use of tampered data can lead to incorrect production decisions and reduced production efficiency, so ensuring the integrity of data is necessary. First, ES can detect any modification of the message sent by SD. Second, DU can detect any modification of the packaged data stored in the CS.
3) *Anonymity:* If the anonymity of the original data is not guaranteed, an attacker can infer private information about the IIoT from multiple sets of original data. Therefore, the anonymity of original data must be ensured, and the real identity of SDs should be hidden.

TABLE I
NOTATIONS AND DEFINITIONS USED

Notations	Definitions
KGC	Key generation center
ES	Edge server
DU	Data user
SD_i	i-th smart device
CS	Cloud server
BC	Blockchain
msk, P_{pub}	System master secret key and public key
ID_{ES}	Real identity of ES
sk_{ES}, PK_{ES}	Secret key and public key of ES
ID_{DU}	Real identity of DU
sk_{DU}, PK_{DU}	Secret key and public key of DU
RID_i	Real identity of SD_i
PID_i	Pseudonym of SD_i
VP_i	Validity period of PID_i
sk_i, PK_i	Secret key and public key of SD_i
ask, APK	Assisted secret key and assisted public key
m_i	Original data
$index$	Index of packaged data
rk	Re-encryption key

2) Functional Goals: Our proposed scheme should achieve the following functional goals.

1) *Data Packaging:* ES can verify a batch of data from SDs and then package the legitimate data.
2) *Storage and Access Authentication:* First, BC can verify the legitimacy of ES's storage request. Second, BC can verify the legitimacy of DU's access request.
3) *On-Chain and Off-Chain Collaborative Storage:* The CS stores the packaged data. The BC holds the metadata corresponding to the packaged data, which can realize effective supervision of the packaged data.

V. PROPOSED SCHEME

This section presents the proposed scheme in detail. And, some notations used in the scheme are listed in Table I.

A. System Initialization

First, KGC executes the $\mathbf{Setup_{sys}}$ algorithm to generate system parameters. When ES/DU submits a registration request with their real identities and the service they want to subscribe to, KGC runs the $\mathbf{Setup_{ES}}$/$\mathbf{Setup_{DU}}$ algorithm to generate corresponding parameters and then sends them to the ES/DU via a secure channel. Finally, KGC stores the public parameter about service into *List*, where the *List* is in the BC. When SD_i submits a registration request, KGC executes the $\mathbf{Setup_{SD}}$ algorithm to generate the corresponding parameters and sends them to SD_i via a secure channel. Finally, KGC deploys smart contracts.

1) *$Setup_{sys}(1^\lambda)$:* On input a security parameter λ, this algorithm first chooses a random number $msk \in \mathbb{Z}_q^*$ as the system master secret key, and computes $P_{\text{pub}} = msk \cdot P$ as the system public key. Then, this algorithm selects some hash functions $H_1 : \mathbb{G} \to \{0, 1\}^*$, $H_2 : \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$, $H_3 : \mathbb{Z}_q^* \times \{0, 1\}^* \to \mathbb{Z}_q^*$, $H_4 : \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$, $H_5 : \{0, 1\}^* \times \mathbb{Z}_q^* \times \mathbb{G} \to \mathbb{Z}_q^*$, $H_6 : \mathbb{Z}_q^* \times \mathbb{Z}_q^* \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$,

$H_7 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$, $H_8 : \{0,1\}^* \times \mathbb{Z}_q^* \times \mathbb{Z}_q^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$, and $H_9 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{Z}_q^* \times \mathbb{Z}_q^* \times \mathbb{Z}_q^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$. Finally, this algorithm outputs $\{msk, P_{\text{pub}}, H_i(i = 1, \ldots, 9)\}$. After executing this algorithm, KGC keeps msk secret and publishes public parameters $\{P_{\text{pub}}, H_i(i = 1, \ldots, 9)\}$ to the IIoT system.

2) *$Setup_{ES}(ID_{ES}, \alpha)$:* On input ES's real identity $ID_{ES} \in \{0,1\}^*$ and service $\alpha \in \{0,1\}^*$, this algorithm first chooses a random number $sk_{ES} \in \mathbb{Z}_q^*$ as the ES's secret key and calculates $PK_{ES} = sk_{ES} \cdot P$ as the ES's public key. Subsequently, this algorithm selects a random number $k \in \mathbb{Z}_q^*$ as the encryption seed associated with the service α, then it selects a random number $w \in \mathbb{Z}_q^*$ and computes $W = w \cdot P$ as the public parameter associated with the service α. Finally, this algorithm outputs $\{sk_{ES}, PK_{ES}, k, w, W\}$. After executing this algorithm, KGC sends $\{sk_{ES}, PK_{ES}, k, w\}$ to ES via a secure channel and stores W in the *List*.

3) *$Setup_{DU}(ID_{DU}, \alpha)$:* On input the DU's real identity $ID_{DU} \in \{0,1\}^*$ and α, this algorithm first chooses a random number $sk_{DU} \in \mathbb{Z}_q^*$ as the DU's secret key and calculates $PK_{DU} = sk_{DU} \cdot P$ as the DU's public key. Then, based on the service α, this algorithm chooses w for the DU. Finally, this algorithm outputs $\{sk_{DU}, PK_{DU}, w\}$. After executing this algorithm, KGC sends $\{sk_{DU}, PK_{DU}, w\}$ to DU via a secure channel.

4) *$Setup_{SD}(msk, ID_{ES}, RID_i)$:* On input SD_i's real identity $RID_i \in \{0,1\}^*$, msk and ID_{ES}, this algorithm chooses a random number $r_i \in \mathbb{Z}_q^*$ and calculates $PID_i = RID_i \oplus H_1(r_i \cdot P_{pub})$ as the SD_i's pseudonym. Then, this algorithm computes $s_i = H_2(PID_i, VP_i, PK_i, PK_{ES})$, where $PK_i = r_i \cdot P$, VP_i is the validity period of PID_i. Subsequently, this algorithm computes $sk_i = r_i + msk \cdot s_i$ as the secret key of SD_i. Finally, this algorithm outputs $\{PID_i, sk_i, PK_i, VP_i\}$. After executing this algorithm, KGC sends $\{PID_i, sk_i, PK_i, VP_i\}$ to SD_i via a secure channel and sends $\{PID_i, VP_i\}$ to ES.

Remark 1: To ensure the anonymity and un-linkability of the original data, an SD is assigned many pseudonyms. And every once in a while, the SD will use another new pseudonym.

B. Original Data Encryption and Signing

In this phase, assuming that the service is α, ES first runs the **AKGen** algorithm to generate the assisted public/secret key pair and sends the assisted public key to SD. Subsequently, SD runs the **Encrypt−I** algorithm to encrypt the original data and executes the **Sign** algorithm to sign the ciphertext. Finally, SD sends the signed message to ES.

1) *$AKGen(k, T)$:* On input k and time period T, this algorithm computes an assisted secret key $ask = H_3(k, T)$ and assisted public key $APK = ask \cdot P$. Finally, this algorithm outputs $\{ask, APK\}$. After executing this algorithm, ES sends APK to SD_i.

2) *$Encrypt - I(m_i, APK)$:* On input original data $m_i \in \{0,1\}^*$ and APK, this algorithm chooses a random number $u_i \in \mathbb{Z}_q^*$. Then, this algorithm calculates ciphertext $c_i = m_i \oplus H_1(u_i \cdot APK)$ and outputs $\{u_i, c_i\}$.

3) *$Sign - I(c_i, u_i, sk_i, PID_i, PK_i, VP_i, PK_{ES}, APK, ts_i)$:* On input $\{c_i, u_i, sk_i, PID_i, PK_i, VP_i, PK_{ES}, APK\}$ and current timestamps ts_i, this algorithm first computes $U_i = u_i \cdot P$ and $s_i^* = H_4(c_i, PID_i, VP_i, ts_i, U_i, PK_i, PK_{ES}, APK)$. Then, the algorithm computes signature $\sigma_i = sk_i + u_i \cdot s_i^*$. Finally, this algorithm outputs $\{\sigma_i, U_i\}$. After executing this algorithm, SD_i sends $\delta_i = (\sigma_i, c_i, U_i, PID_i, PK_i, APK, ts_i)$ to ES.

C. Message Verification

Upon receiving the final message $\delta_i = (\sigma_i, c_i, U_i, PID_i, PK_i, APK, ts_i)$ from SD, ES first checks whether ts_i is fresh. If not, the final message δ_i is discarded. Then, ES checks whether the VP_i corresponding to PID_i is valid. If not, ES discards δ_i. Subsequently, ES classifies the final messages according to services. Assume there are n final messages corresponding to α, which are $(\sigma_1, c_1, U_1, PID_1, PK_1, APK, ts_1)$, $(\sigma_2, c_2, U_2, PID_2, PK_2, APK, ts_2)$, $\ldots$, $(\sigma_n, c_n, U_n, PID_n, PK_n, APK, ts_n)$. Finally, ES executes the **Verify** algorithm to verify the message.

1) *$Verify(\sigma_i, c_i, U_i, PID_i, PK_i, VP_i, PK_{ES}, APK, ts_i)$:* On input $(\sigma_1, c_1, U_1, PID_1, PK_1, VP_1, PK_{ES}, APK, ts_1)$, $(\sigma_2, c_2, U_2, PID_2, PK_2, VP_2, PK_{ES}, APK, ts_2)$, $\ldots$, $(\sigma_n, c_n, U_n, PID_n, PK_n, VP_n, PK_{ES}, APK, ts_n)$, this algorithm computes $s_i = H_2(PID_i, VP_i, PK_i, PK_{ES})$ and $s_i^* = H4(c_i, PID_i, VP_i, ts_i, U_i, PK_i, PK_{ES}, APK)$. Subsequently, to ensure nonrepudiation, we use the small exponential test technique [38]. Therefore, the algorithm chooses a vector $v = \{v_1, v_2, \ldots, v_n\}$ and verifies the batch data by checking the following equation:

$$\left(\sum_{i=1}^{n}(v_i \cdot \sigma_i)\right) \cdot P = \sum_{i=1}^{n}(v_i \cdot PK_i) + \left(\sum_{i=1}^{n}(v_i \cdot s_i)\right) \cdot P_{\text{pub}} + \sum_{i=1}^{n}\big(v_i \cdot s_i^* \cdot U_i\big). \tag{1}$$

If the equation holds, this algorithm outputs 1; otherwise, outputs 0.

The correctness (1) is as follows:

$$\begin{aligned}\left(\sum_{i=1}^{n}(v_i \cdot \sigma_i)\right) \cdot P &= \sum_{i=1}^{n}\big(v_i \cdot (sk_i + u_i \cdot s_i^*)\big) \cdot P \\ &= \sum_{i=1}^{n}\big(v_i \cdot ((r_i + msk \cdot s_i) + u_i \cdot s_i^*)\big) \cdot P \\ &= \sum_{i=1}^{n}\big(v_i \cdot ((r_i \cdot P + s_i \cdot msk \cdot P) + s_i^* \cdot u_i \cdot P)\big) \\ &= \sum_{i=1}^{n}\big(v_i \cdot ((PK_i + s_i \cdot P_{\text{pub}}) + s_i^* \cdot U_i)\big) \\ &= \sum_{i=1}^{n}(v_i \cdot PK_i) + \left(\sum_{i=1}^{n}(v_i \cdot s_i)\right) P_{\text{pub}} + \sum_{i=1}^{n}\big(v_i \cdot s_i^* \cdot U_i\big). \end{aligned} \tag{2}$$

D. Packaged Data Generation

After the verification, ES packages the valid data by computing $sd = (U_1||c_1)||(U_2||c_2)||\ldots||(U_n||c_n)$, then the ES stores the packaged data in CS and generates the packaged data index *index*. Subsequently, ES executes the **Encrypt − II** algorithm to generate ciphertext k_{ES} and corresponding assisted parameters $\{h, d, h_A, s_A\}$.

1) *Encrypt* − *II*$(ask, T, sd, w, W, k, index, PK_{ES})$*:* On input $(ask, T, sd, w, W, k, PK_{ES})$ and packaged data index *index*, this algorithm computes $h = H_5(sd, ask, W)$. Then, this algorithm calculates $d = H_6(ask, w, k)$, $D = d \cdot P$, and $k_{ES} = (ask||index) \oplus H_1(d \cdot PK_{ES})$. Subsequently, this algorithm computes $h_A = H_8(index, ask, h, PK_{ES}, D, W)$ and $s_A = sk_{ES} + d \cdot h_A$. Finally, this algorithm outputs $\{h, d, k_{ES}, h_A, s_A\}$.

E. Storage and Access Authentication

When ES wants to store data to BC, it first executes the **Sign − II** algorithm to generate signature σ_{upload} and sends the storage request $sr = (\sigma_{\text{upload}}, PK_{ES}, W, T, h, k_{ES}, h_A, s_A, X_i)$ to BC. Once receives the *sr*, BC checks whether W exists in the *List*. If it exists, BC executes the **Test − I** algorithm to determine whether the storage request is valid. If so, BC stores the metadata $metadata = (PK_{ES}, W, T, h, k_{ES}, h_A, s_A)$. When DU wants to access packaged data, it first executes the **Sign − III** algorithms to generate signature σ_{download} and sends the access request $ar = (\sigma_{\text{download}}, PK_{ES}, PK_{DU}, T, W, Y_i)$ to BC. Once receives the *ar*, BC checks whether W exists in the *List*. If it exists, BC executes the **Test − II** algorithm to determine whether the access request is valid. If so, BC sends the corresponding request $\{W, T, PK_{DU}\}$ to the ES.

1) *Sign* − *II*$(sk_{ES}, w, PK_{ES}, W, h, k_{ES}, h_A, s_A, T)$*:* On input $(sk_{ES}, w, PK_{ES}, W, h, k_{ES}, h_A, s_A, T)$, the algorithm selects a random number $x_i \in \mathbb{Z}_q^*$, computes signature $\sigma_{\text{upload}} = sk_{ES} + w + x_i \cdot H_9(T, k_{ES}, h, h_A, s_A, PK_{ES}, W, X_i)$ and outputs the signature σ_{upload} and X_i, where $X_i = x_i \cdot P$.
2) *Test* − *I*$(\sigma_{\text{upload}}, PK_{ES}, W, T, h, k_{ES}, h_A, s_A, X_i)$*:* On input $(\sigma_{\text{upload}}, PK_{ES}, W, T, h, k_{ES}, h_A, s_A, X_i)$, this algorithm checks whether the following equation holds:
$$\sigma_{\text{upload}} \cdot P = PK_{ES} + W + H_9(T, k_{ES}, h, h_A, s_A, PK_{ES}, W, X_i) \cdot X_i. \quad (3)$$
If holds, this algorithm returns 1; otherwise, returns 0.
3) *Sign* − *III*$(sk_{DU}, T, PK_{DU}, PK_{ES}, w, W)$*:* On input $(sk_{DU}, T, PK_{DU}, PK_{ES}, w, W)$, the algorithm selects a random number $y_i \in \mathbb{Z}_q^*$, computes signature $\sigma_{\text{download}} = sk_{DU} + w + y_i \cdot H_7(T, PK_{DU}, PK_{ES}, W, Y_i)$ and outputs the signature σ_{download} and Y_i, where $Y_i = y_i \cdot P$.
4) *Test* − *II*$(\sigma_{\text{download}}, PK_{ES}, PK_{DU}, T, W, Y_i)$*:* On input $(\sigma_{\text{download}}, PK_{ES}, PK_{DU}, T, W, Y_i)$, this algorithm checks whether the following equation holds:
$$\sigma_{\text{download}} \cdot P = PK_{DU} + W + H_7(T, PK_{DU}, PK_{ES}, W, Y_i) \cdot Y_i. \quad (4)$$
If holds, this algorithm returns 1; otherwise, returns 0.

Remark 2: The blockchain has limited throughput and is difficult to store large-scale data. If many SDs interact directly with the blockchain and store data, it may impose a huge storage burden and operational latency on the blockchain. In the proposed scheme, ES packages data from SDs and stores them in the CS. Then, the ES stores the metadata corresponding to packaged data in the BC. On the one hand, the data packaging mechanism reduces the number of interactions between ES and BC significantly. If there is no packing mechanism, the number of interactions between ES and BC increases with the number of original data. On the other hand, the length of metadata is shorter than the length of packaged data. BC only stores metadata rather than packaged data, effectively reducing storage pressure.

F. Re-Encryption Key Generation and Re-Encryption

ES first chooses d based on $\{W, T\}$ and then runs the **ReKeyGen** algorithm to get the re-encryption key *rk*. Subsequently, ES sends *rk* to BC. Once BC receives *rk*, it runs the **ReEncrypt** algorithm to get ciphertext k_{DU}. Finally, BC sends access response $\{k_{DU}, s_A, h_A, h\}$ to DU.

1) *ReKeyGen*(d, PK_{ES}, PK_{DU})*:* On input (d, PK_{ES}, PK_{DU}), this algorithm computes re-encryption key $rk = H_1(d \cdot PK_{ES}) \oplus H_1(d \cdot PK_{DU})$ and then outputs the *rk*.
2) *ReEncrypt*(k_{ES}, rk)*:* On input (k_{ES}, rk), this algorithm computes $k_{DU} = k_{ES} \oplus rk$ and then outputs it.

G. Re-Decryption and Decryption

After receiving the access response from BC, DU first runs the **ReDecrypt** algorithm to obtain *ask* and *index*, then queries the packaged data *sd* through *index*, and finally runs the **Decrypt** algorithm to obtain the original data m_i.

1) *ReDecrypt*$(sk_{DU}, k_{DU}, PK_{ES}, s_A, h_A, h, W)$*:* On input $(sk_{DU}, k_{DU}, PK_{ES}, s_A, h_A, W)$, this algorithm computes $D = (s_A \cdot P - PK_{ES})/h_A$. Then, this algorithm calculates $ask||index = k_{DU} \oplus H_1(sk_{DU} \cdot D)$. Finally, this algorithm checks whether the equation $h_A = H_8(index, ask, h, PK_{ES}, D, W)$ holds, if it does, it outputs *ask* and *index*. Otherwise, it outputs $\perp$.
2) *Decrypt*(h, sd, ask, W)*:* On input (h, sd, ask, W), this algorithm first checks whether the equation $h = H_5(sd, ask, W)$ holds. If not, it proves that the packaged data has been tampered with and this algorithm outputs $\perp$. Otherwise, this algorithm computes $c_i \oplus H_1(ask \cdot U_i)$ to obtain the original data m_i and outputs it.

VI. Security Proof and Analysis

The integrity of messages sent by SD is based on the DL problem. The integrity of packaged data is based on the collision resistance of the hash function. The confidentiality of original data, assisted secret key, and packaged data index relies on the CDH problem and the collision resistance of the hash function. Therefore, the way of proving the original data confidentiality is similar to the way of proving the assisted secret key and packaged data index confidentiality. For convenience, we focus on the security proof of assisted secret key and packaged data index confidentiality. We define a ciphertext indistinguishability game, which is an interaction between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.

A. Security Proof

In the proof, we use $m = (ask||index)$ to represent the plaintext m. The ciphertext indistinguishability game performs as follows.

1) Setup: The challenger $\mathcal{C}$ first executes the $\mathbf{Setup_{sys}}$ algorithm to generate system parameters. Then, it runs $\mathbf{Setup_{ES}}$ and $\mathbf{Setup_{DU}}$ algorithms to generate the corresponding parameters $\{sk_{ES}, PK_{ES}, k, w\}$ and $\{sk_{DU}, PK_{DU}, w\}$, respectively. Finally, $\mathcal{C}$ sends public system parameters to $\mathcal{A}$.

2) Phase 1: The adversary $\mathcal{A}$ adaptive launches some queries to $\mathcal{C}$ as the following oracle.

1) *Ciphertext oracle $\mathcal{O}$:* Given a plaintext message m, the oracle executes the **Encrypt − II** algorithm to generate the ciphertext k_{ES}.

3) Challenge: $\mathcal{A}$ selects two plaintext message (m_0, m_1) and sends them to $\mathcal{C}$. The $\mathcal{C}$ chooses a random bit $c \in \{0, 1\}$ and then executes the **Encrypt − II** algorithm to generate the corresponding ciphertext $\{k_{ES}, h_A, s_A\}$. Finally, $\mathcal{C}$ sends $\{k_{ES}, h_A, s_A, PK_{ES}\}$ to $\mathcal{A}$.

4) Phase 2: The adversary $\mathcal{A}$ can launch queries to the $\mathcal{O}$. Noting that $\mathcal{A}$ can not submit m_0 or m_1 to the oracle $\mathcal{O}$.

5) Guess: $\mathcal{A}$ outputs a bit $c' \in \{0, 1\}$. If $c' = c$, then the $\mathcal{A}$ wins the game.

The $\mathcal{A}$'s advantage winning the above game is defined as

$$Adv_{\mathcal{A}}^{IND}(\lambda) = \left|Pr[c' = c] - \frac{1}{2}\right|.$$

6) Theorem 1: In the random oracle model, the proposed scheme can achieve ciphertext indistinguishability if the solution to the CDH problem is negligible and the hash function is collision resistant.

7) Proof: We demonstrate the ciphertext indistinguishability through a series of related games. In the last game, we prove that the ciphertext and plaintext messages are independent. ■

Game 0: The game is the same as the ciphertext indistinguishability game. The challenger $\mathcal{C}$ first generates system parameters $\{P, H_1, H_3, H_5, H_6, H_8\}$ and two public/secret key pairs $(PK_{ES} = sk_{ES} \cdot P, sk_{ES})$ and $(PK_{DU} = sk_{DU} \cdot P, sk_{DU})$. Then, $\mathcal{C}$ publishes the public data $(P, H_1, H_3, H_5, H_6, H_8, PK_{ES}, PK_{DU})$ and keeps secret keys (sk_{ES}, sk_{DU}). To response the ciphertext oracle query, for $m = (ask||index)$, $\mathcal{C}$ computes $h = H_5(sd, ask, W)$, $d = H_6(ask, w, k)$, $D = d \cdot P$, and $ask = H_3(k, T)$. Then, $\mathcal{C}$ generates the ciphertext $k_{ES} = m \oplus H_1(d \cdot PK_{ES})$ and computes $h_A = H_8(index, ask, h, PK_{ES}, D, W)$, $s_A = sk_{ES} + d \cdot h_A$. This process simulates $\mathcal{A}$'s ability to require the corresponding ciphertext. After the above series of queries, $\mathcal{A}$ guesses the challenge ciphertext and finally wins this game with the advantage is

$$Adv_{\mathcal{A}}^{\text{Game0}}(\lambda) = Adv_{\mathcal{A}}^{IND}(\lambda).$$

Game 1: The game is the same as **Game 0** except it uses a secure hash function H_1. We model the hash function as the random oracle. For the H_1 query, $\mathcal{C}$ presets a map Map_{H_1}. When $\mathcal{A}$ makes an H_1 query with $< x >$, $\mathcal{C}$ checks whether the Map_{H_1} has the key $< x >$. If so, $\mathcal{C}$ returns the corresponding value to $\mathcal{A}$. Otherwise, $\mathcal{C}$ random selects a value y and sets $Map_{H_1}(< x >) = y$. Because the hash function is secure, it is indistinguishable between **Game 1** and the above game. $\mathcal{A}$ wins this game with the advantage is

$$Adv_{\mathcal{A}}^{\text{Game1}}(\lambda) = Adv_{\mathcal{A}}^{\text{Game0}}(\lambda).$$

Game 2: The game is similar to **Game 1**, but instead of calculating $k_{ES} = m \oplus H_1(d \cdot PK_{ES})$ to get the corresponding ciphertext, this game selects a random string $S \in \{0, 1\}^*$ and then calculates $k'_{ES} = m \oplus S$ to get the corresponding ciphertext k'_{ES}. In **Game 2**, given a CDH problem instance $\{P, sk_{ES} \cdot P, d \cdot P\}$. The adversary $\mathcal{A}$ does not know (sk_{ES}, d) and cannot solve the CDH problem with nonnegligible probability, so it cannot distinguish between k'_{ES} and k_{ES}. Therefore, $\mathcal{A}$ wins this game with the advantage is

$$|Adv_{\mathcal{A}}^{\text{Game2}}(\lambda) - Adv_{\mathcal{A}}^{\text{Game1}}(\lambda)| \leq Adv_{\mathcal{A}}^{CDH}(\lambda),$$

where the $Adv_{\mathcal{A}}^{CDH}(\lambda)$ indicates the advantage of $\mathcal{A}$ solves the CDH problem within polynomial time. Solving the CDH problem is hard, so the $Adv_{\mathcal{A}}^{CDH}(\lambda)$ is negligible.

Game 3: The game is similar to **Game 2**, but does not obtain (s_A, k_{ES}) by computing $s_A = sk_{ES} + d \cdot H_8(index, ask, h, PK_{ES}, D, W)$ and $k_{ES} = m \oplus S$. Instead, the challenger $\mathcal{C}$ randomly chooses a number $s'_A \in \mathbb{Z}_q^*$ and a string $k''_{ES} \in \{0, 1\}^*$. On the one hand, the generation of s_A needs some random numbers (e.g., k and ω) and the hash function is collision resistant. Therefore, in the view of $\mathcal{A}$, s_A and s'_A are indistinguishable. On the other hand, S is randomness, so in the view of $\mathcal{A}$, k''_{ES} and k'_{ES} are indistinguishable. In summary, the $\mathcal{A}$ wins this game with the advantage is

$$Adv_{\mathcal{A}}^{\text{Game3}}(\lambda) = Adv_{\mathcal{A}}^{\text{Game2}}(\lambda).$$

Through the above game, we find that the challenge ciphertext is completely independent of the plaintext m_c, so the advantage of $\mathcal{A}$ winning **Game 3** is

$$Adv_{\mathcal{A}}^{\text{Game3}}(\lambda) = |(1/2) - (1/2)| = 0$$

and we can obtain

$$Adv_{\mathcal{A}}^{IND}(\lambda) \leq Adv_{\mathcal{A}}^{CDH}(\lambda)$$

according to Definition 2, we know that the advantage $Adv_{\mathcal{A}}^{CDH}(\lambda)$ is negligible. Therefore, under the random oracle model, our proposed scheme can satisfy the confidentiality of assisted secret key and packaged data index.

B. Security Analysis

1) Confidentiality: On the one hand, before sending the original data, SD executes the **Encrypt−I** algorithm to encrypt the original data, and the ciphertext can only be decrypted by legitimate DUs. On the other hand, assisted secret key and packaged data index are encrypted by ES, and only authorized DUs can execute the **ReDecrypt** algorithm to obtain them. Therefore, our scheme can ensure data confidentiality.

2) Integrity: On the one hand, our scheme can guarantee the integrity of messages sent by SD because solving the DL problem is hard. Hence, the ES can check the message integrity by verifying whether $\delta_i \cdot P = PK_i + s_i \cdot P_{\text{pub}} + s_i^* \cdot U_i$ holds. On the other hand, our scheme can ensure the integrity of packaged data because the hash function is collision resistant. Hence, the DU can check the integrity of

TABLE II
COMPARISON OF SECURITY AND FUNCTIONALITY FEATURES

	[20]	[21]	[19]	Ours
Confidentiality	✓	✓	✓	✓
Integrity	✓	✓	✓	✓
Anonymity	✓	×	✓	✓
Data packaging	×	×	×	✓
Storage authentication	✓	×	✓	✓
Access authentication	×	✓	✓	✓
On-chain and off-chain collaborative storage	✓	×	✓	✓

the packaged data by verifying whether the equation $h = H_5(sd, ask, W)$ holds.

3) Anonymity: In the proposed scheme, SD hides its real identity RID_i in the pseudonym PID_i, where $PID_i = RID_i \oplus H_1(r_i \cdot P_{\text{pub}})$. To obtain the RID_i from PID_i, the malicious attacks need to compute $r_i \cdot msk \cdot P$ from $R_i = r_i \cdot P$ and $P_{\text{pub}} = msk \cdot P$. However, solving the CDH problem is hard, so our scheme can guarantee data anonymity.

C. Comparison of Security and Functionality Features

From Table II, we can see that our proposed scheme has advantages over other related schemes [19], [20], [21].

VII. PERFORMANCE EVALUATION

A. Experimental Settings

1) Related Schemes Setting: To make the performance evaluation fairer, we introduce three other blockchain-based data-sharing schemes into the data-sharing framework we considered. The specific descriptions are as follows.

1) In Manzoor et al.'s scheme [20], SD generates original data and packaged data. CS stores the packaged data, and BC performs proxy re-encryption operations.
2) In Chen et al.'s scheme [21], SD generates original data and packaged data. BC stores the packaged data and performs re-encryption operations.
3) In Lu et al.'s scheme [19], SD generates original data and packaged data. BC performs proxy re-encryption and CS stores the packaged data.

2) Experimental Environment Setting: We use the Miracl Core [39] cryptography library and choose the BLS12381 type curve, which achieves the security of 128 bits. ES and DU operations are executed on a PC running Ubuntu 18.04.3 with an Intel Core i5-7500 CPU @3.4 GHz and 16-GB RAM, while SD operations are performed on a Raspberry Pi 4 running Debian GNU/Linux 11 with a 1.5-GHz CPU and 4-GB RAM. We use hyperledger fabric [40] to build a blockchain platform consisting of ten PCs. Each PC has an Intel Core i7-11700 CPU @2.50 GHz and 16-GB RAM, all running Ubuntu 18.04.3. One PC serves as the client application node, one serves as the orderer node, and the remaining eight serve as peer nodes.

B. Computation Cost Analysis

1) Theoretical Analysis: We set up a total of n SDs subscribing to the same service and consider the following time-consuming operations.

1) sm_1: A scale multiplication operation $x \cdot P_1$, where $x \in \mathbb{Z}_q^*$ and $P_1 \in \mathbb{G}_1$.
2) sm_2: A scale multiplication operation $x \cdot P_2$, where $x \in \mathbb{Z}_q^*$ and $P_2 \in \mathbb{G}_2$.
3) ge_1: An exponential operation P_1^x, where $x \in \mathbb{Z}_q^*$ and $P_1 \in \mathbb{G}_1$.
4) ge_2: An exponential operation P_2^x, where $x \in \mathbb{Z}_q^*$ and $P_2 \in \mathbb{G}_2$.
5) bp: A bilinear pairing operation $e(g_1, g_2)$, where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.
6) ge_t: An exponential operation P_t^x, where $x \in \mathbb{Z}_q^*$ and $P_t \in \mathbb{G}_T$.

In our scheme, SD needs $1sm_1$ to encrypt the original data and $1sm_1$ to sign the ciphertext, so the total time-consuming operations are $2sm_1$. ES requires $(n+2)sm_1$ for batch verification, $2sm_1$ when generating packaged data, $1sm_1$ when generating a storage request, and $2sm_1$ when generating re-encryption keys, so the total time-consuming operations performed by ES are $(n+7)sm_1$. For BC, it mainly executes **Test − I** and **Test − II** algorithms, so the total time-consuming operations required are $4sm_1$. DU needs $1sm_1$ to execute the **Sign − III** algorithm, $3sm_1$ to execute the **ReDecrypt** algorithm, and nsm_1 to execute the **Decrypt** algorithm. Therefore, the total time-consuming operations required by the DU are $(n+4)sm_1$.

We calculate the time-consuming operations for other schemes using the same method, and the results are shown in Table III. In our scheme, the blockchain needs to perform one storage authentication, one proxy re-encryption, and one access authentication regardless of the number of SDs. In [20], the blockchain needs to execute n proxy re-encryption. In [19], the blockchain needs to perform n storage authentications, n proxy re-encryption, and one access authentication. In [21], the blockchain needs to perform n access authentications and n proxy re-encryption.

2) Off-Chain Simulation Result: In Fig. 3(a), we can see that in our scheme, the computation overhead of SD is about 2.745 ms when the number of SDs is 1. The computational cost of SD remains relatively constant even as the number of SDs increases. Furthermore, it is evident that our proposed scheme incurs the lowest computational overhead. When the number of SDs is 1, we obtain the computational overhead of SD in our scheme is 2.745 ms, in [20] is 5.804 ms, in [21] is 11.333 ms, and in [19] is 78.052 ms. Therefore, the computational overhead of SD in our scheme is about $2.745/5.804 \approx 47.29\%$ in [20], about $2.745/11.333 \approx 24.22\%$ in [21], and about $2.745/78.052 \approx 3.52\%$ in [19]. The reason is that in [20], SD encrypts the original data and generates the re-encryption key. The SD in [21] needs to generate an authorization trapdoor. In [19], SD uses many multiplication and pairing operations with high computational overhead during the signing.

In Fig. 3(b), we can see that in [20] and [21], the ES does not have any computational overhead. Because in both schemes, the SD communicates directly with the BC. When the number of SDs is 1, the computational overhead of ES in our scheme is 2.585 ms, and in [19] is 0.274 ms. As the number of SDs increases, our scheme and [19] will increase in computational overhead. Although the computational overhead

TABLE III
COMPARISON OF TIME-CONSUMING CRYPTOGRAPHIC OPERATIONS

Scheme	SD	ES	BC	DU
Manzoor *et al.* [20]	$5sm_1$	-	-	$3nsm_1$
Chen *et al.* [21]	$7sm_1 + 1sm_2$	-	$2nbp$	$8nsm_1 + 4nsm_2$
Lu *et al.* [19]	$19ge_1 + 3ge_2 + 4ge_t + 4bp$	nge_1	$(10n+6)ge_1 + (2n+2)ge_2 + (5n+5)ge_t + (11n+11)bp$	$(4n+13)ge_1 + 3ge_2 + 4ge_t + 4bp$
Ours	$2sm_1$	$(n+7)sm_1$	$4sm_1$	$(n+4)sm_1$

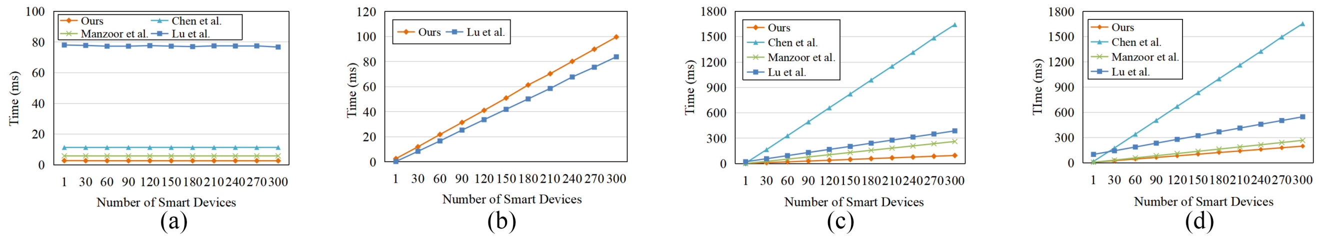


Fig. 3. Off-chain computation cost comparison. (a) Computation cost on SD. (b) Computation cost on ES. (c) Computation cost on DU. (d) Total computation cost.

of ES in our scheme is higher than that in [19], it can be seen from Fig. 3(d) that the total computational overhead of our proposed scheme is lower than that of [19]. Because in [19], the ES is used to trace the SD, and more operations are performed by the SD.

In Fig. 3(c), we can see that as the number of SDs increases, the computational overhead of DU also increases. When the number of SDs is 1, the computational overhead of DU is 1.590 ms in our scheme, is 5.510 ms in [21], is 0.871 ms in [20], and is 21.831 ms in [19]. Although the computational overhead of DU in our scheme is higher than that in [20] when the number of SDs is 1, the computational overhead of DU in our scheme is lower than that in [20] when the number of SDs exceeds 2. The reason is that in our proposed scheme, ES uses the packaging mechanism to generate packaged data, DU only needs to verify the assisted secret key once, and the operation involves only scalar multiplication on $\mathbb{G}_1$. In [20], DU requires n data verification. In [21], DU needs to generate an authorization trapdoor for each SD, so a total of n authorization trapdoors need to be generated, which brings a huge computation overhead. In [19], the operations performed by the DU contain many time-consuming operations.

In Fig. 3(d), we find that when the number of SDs is 1, the computational overhead of our scheme is higher than that in [20]. Still, when the number of SDs exceeds 2, the computational overhead in our scheme becomes lowest. As the number of SDs increases, our scheme's computational overhead growth rate is minimized. When the number of SDs is 300, the computation overhead of our scheme is 197.992 ms, it is about $197.992/267.116 \approx 74.12\%$ in [20], is about $197.992/1655.046 \approx 11.96\%$ in [21], and is about $197.992/547.281 \approx 36.18\%$ in [19].

3) On-Chain Simulation Result: We consider the following operations to evaluate the performance of various schemes in the blockchain.

1) *Our-SA:* The storage authentication in our proposed scheme.
2) *Our-AA:* The access authentication in our proposed scheme.
3) *Chen-AA:* The access authentication in Chen et al.'s scheme [21].
4) *Lu-SA:* The storage authentication in Lu et al.'s scheme [19].
5) *Lu-AA:* The access authentication in Lu et al.'s scheme [19].
6) *Our-RE:* The re-encryption in our proposed scheme.
7) *Chen-RE:* The re-encryption in Chen et al.'s scheme [21].
8) *Lu-RE:* The re-encryption in Lu et al.'s scheme [19].
9) *Manzoor-RE:* The re-encryption in Manzoor et al.'s scheme [20].
10) *Write:* The write operation.
11) *Query:* The query operation.

First, from Fig. 4(a), we can see that when the send rate is within 180 transactions per second (TPS), the time delay of *Our-SA* fluctuates insignificantly, which is about 0.08 s. When the send rate exceeds 180 TPS, the time delay starts to rise significantly, and when the send rate reaches 330 TPS, the time delay is 32.39 s. In Fig. 4(a), we find that *Our-AA* has the same trend of latency variation as *Our-SA*, which is due to the fact that in our scheme, the storage and access authentication takes the same time-consuming operations, both being $2sm_1$. Second, we find that when the send rates are the same, *Chen-AA, Lu-SA,* and *Lu-AA* have higher time delays than in *Our-SA* and *Our-AA*. For example, when the send rate is 1 TPS, the time required for *Our-SA* is equal to that required for *Our-AA*, 0.04-s less than *Chen-AA*, 0.25-s less than *Lu-SA*, and 0.24-s less than *Lu-AA*. In addition, as the send rate increases, the delay for *Chen-AA, Lu-SA*, and *Lu-AA* increases, and the growth rate is greater than that of *Our-SA* and *Our-AA*. This is because there are more time-consuming operations in *Chen-AA, Lu-SA*, and *Lu-AA*.

From Fig. 4(b), we can see that as the send rate increases, the throughput of executing *Our-SA* and *Our-AA* increases accordingly. When the send rate reaches 210, their throughput

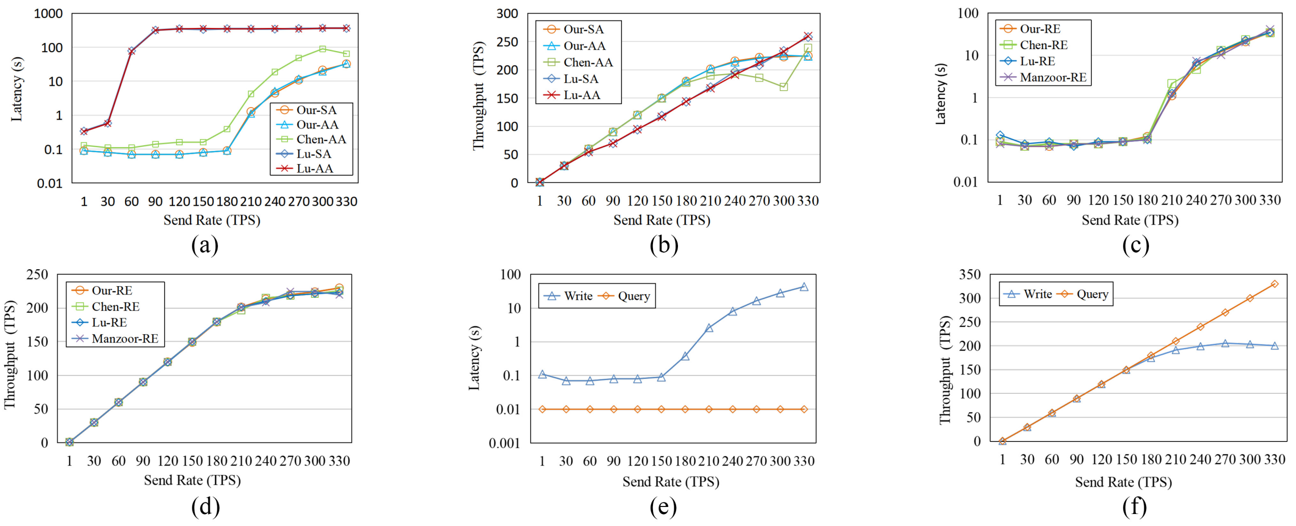


Fig. 4. On-chain performance comparison. (a) Time latency of storage and access authentication. (b) Throughput of storage and access authentication. (c) Time latency of proxy re-encryption. (d) Throughput of proxy re-encryption. (e) Time latency between write and query. (f) Throughput between write and query.

gradually reaches saturation. Also, we find that the throughput of running *Chen-AA, Lu-SA*, and *Lu-AA* reaches saturation faster than executing *Our-SA* and *Our-AA*. This is because there are time-consuming operations in *Chen-AA, Lu-SA*, and *Lu-AA*, which will consume the resources of the blockchain platform more. It is worth noting that when the send rate reaches around 330 TPS, the system throughput for executing *Chen-AA, Lu-SA*, and *Lu-AA* is higher than that for executing *Our-SA* and *Our-AA*. By observing the experimental results, we find that when the sending rate is 330 TPS, the percentage of failed transactions to the total transactions is 0%, 0%, 7.5%, 98.6%, and 97.9% when executing *Our-SA, Our-AA, Chen-AA, Lu-SA*, and *Lu-AA*, respectively. It means that the blockchain platform has limited resources and cannot successfully process *Chen-AA, Lu-SA*, and *Lu-AA* in a timely manner.

Combining Fig. 4(c) and (d), we find that the time latency and throughput of the four re-encryption operations are close. From Fig. 4(c), we can see that when the send rate is within 180 TPS, all four proxy re-encryption latencies are at a low level. When the send rate exceeds 180 TPS, the time of the four re-encryption operations starts to rise significantly. When the send rate is 330 TPS, the time of executing *Our-RE* is 34.44 s, *Chen-RE* is 34.09 s, *Lu-RE* is 34.81 s, and *Manzoor-RE* is 41.09 s. From Fig. 4(d), we find that the throughput of performing *Our-RE, Chen-RE, Lu-RE*, and *Manzoor-RE* increases as the send rate increases, and the throughput of performing these four operations begins to saturate when the send rate exceeds 210 TPS.

Write and query operations are involved in three other related schemes and our scheme. After testing, the results are shown in Fig. 4(e) and (f). With these two figures, we find that write operations are more time consuming than query operations, and the throughput gradually reaches saturation when the send rate exceeds 210 TPS. In addition, we find that the query operation latency stays low level (0.01 s) when the send rate increases. This is because each time the data is queried, the peer node retrieves the data from the local copy.

Insight: To support more services, we need to pay attention to the following two points.

1) Operations in the blockchain need to be lightweight, which can increase the throughput while ensuring that transactions are executed successfully.
2) Entities should minimize their interactions with the blockchain, thus saving the resources of the blockchain.

Through experimental analysis, we find that in our scheme, operations in the blockchain are lightweight. In addition, the scheme we design supports data packaging, which reduces the interaction number between ES/DU and blockchain. Therefore, our scheme is more suitable for IIoT environments than other related schemes.

VIII. Conclusion

In this study, we aim to achieve efficient and secure data sharing in a blockchain-enabled IIoT system. First, we design a lightweight data-sharing scheme. In this scheme, we consider storage and access authentication and combine them with proxy re-encryption techniques to realize secure data sharing. Second, To reduce the blockchain's storage overhead and computation overhead, the scheme achieves on-chain and off-chain collaborative storage and supports data packaging. Finally, the security analysis demonstrates that the proposed scheme can satisfy the security requirements of the IIoT. The performance evaluation results show that the proposed scheme is feasible and suitable for blockchain-enabled IIoT environments. In the future, we will design a practical data-sharing scheme for mobile SDs.

Acknowledgment

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this article.

References

[1] Y. He et al., "Two-timescale resource allocation for automated networks in IIoT," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 7881–7896, Oct. 2022.
[2] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in Industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

[3] F. Wang, J. Cui, Q. Zhang, D. He, C. Gu, and H. Zhong, "Blockchain-based lightweight message authentication for edge-assisted cross-domain Industrial Internet of Things," *IEEE Trans. Dependable Secure Comput.*, early access, Jun. 15, 2023, doi: 10.1109/TDSC.2023.3285800.

[4] L. Liu and W. Yu, "A D2D-based protocol for ultra-reliable wireless communications for industrial automation," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5045–5058, Aug. 2018.

[5] Y. Jiang, Y. Zhong, and X. Ge, "IIoT data sharing based on blockchain: A multileader multifollower Stackelberg game approach," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4396–4410, Mar. 2022.

[6] Q. Li, J. Chen, M. Cheffena, and X. Shen, "Channel-aware latency tail taming in industrial IoT," *IEEE Trans. Wireless Commun.*, vol. 22, no. 9, pp. 6107–6123, Sep. 2023.

[7] C.-K. Chu, S. S. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 468–477, Feb. 2014.

[8] J. Cui, J. Lu, H. Zhong, Q. Zhang, C. Gu, and L. Liu, "Parallel key-insulated multiuser searchable encryption for Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4875–4883, Jul. 2022.

[9] C. Huang, D. Liu, J. Ni, R. Lu, and X. Shen, "Achieving accountable and efficient data sharing in Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 1416–1427, Feb. 2021.

[10] Z. Qin, H. Xiong, S. Wu, and J. Batamuliza, "A survey of proxy re-encryption for secure data sharing in cloud computing," *IEEE Trans. Services Comput.*, early access, Apr. 6, 2016, doi: 10.1109/TSC.2016.2551238.

[11] J. Shen, H. Yang, P. Vijayakumar, and N. Kumar, "A privacy-preserving and untraceable group data sharing scheme in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2198–2210, Jul./Aug. 2022.

[12] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2374–2385, Aug. 2016.

[13] S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 2101–2113, 2018.

[14] J. Cui, X. Zhang, H. Zhong, J. Zhang, and L. Liu, "Extensible conditional privacy protection authentication scheme for secure vehicular networks in a multi-cloud environment," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1654–1667, 2020.

[15] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.

[16] Y. Wu, H.-N. Dai, and H. Wang, "Convergence of blockchain and edge computing for secure and scalable IIoT critical infrastructures in industry 4.0," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2300–2317, Feb. 2021.

[17] J. Sunny, N. Undralla, and V. M. Pillai, "Supply chain transparency through blockchain-based traceability: An overview with demonstration," *Comput. Ind. Eng.*, vol. 150, Dec. 2020, Art. no. 106895. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835220305829

[18] W. Sun, J. Liu, Y. Yue, and P. Wang, "Joint resource allocation and incentive design for blockchain-based mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 9, pp. 6050–6064, Sep. 2020.

[19] J. Lu, J. Shen, P. Vijayakumar, and B. B. Gupta, "Blockchain-based secure data storage protocol for sensors in the Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5422–5431, Aug. 2022.

[20] A. Manzoor, A. Braeken, S. S. Kanhere, M. Ylianttila, and M. Liyanage, "Proxy re-encryption enabled secure and anonymous IoT data sharing platform based on blockchain," *J. Netw. Comput. Appl.*, vol. 176, Feb. 2021, Art. no. 102917. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520303763

[21] B. Chen, D. He, N. Kumar, H. Wang, and K.-K. R. Choo, "A blockchain-based proxy re-encryption with equality test for vehicular communication systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2048–2059, Jul.–Sep. 2021.

[22] A. Hari, M. Kodialam, and T. V. Lakshman, "ACCEL: Accelerating the Bitcoin blockchain for high-throughput, low-latency applications," in *Proc. IEEE IEEE Conf. Comput. Commun.*, 2019, pp. 2368–2376.

[23] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey on the scalability of blockchain systems," *IEEE Netw.*, vol. 33, no. 5, pp. 166–173, Sep./Oct. 2019.

[24] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Jan. 2019.

[25] K. Miyachi and T. K. Mackey, "hOCBS: A privacy-preserving blockchain framework for healthcare data leveraging an on-chain and off-chain system design," *Inf. Process. Manage.*, vol. 58, no. 3, 2021, Art. no. 102535. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457321000431

[26] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology*, R. Cramer, Ed. Berlin, Germany: Springer, 2005, pp. 457–473.

[27] Z. Kang, J. Li, J. Shen, J. Han, Y. Zuo, and Y. Zhang, "TFS-ABS: Traceable and forward-secure attribute-based signature scheme with constant-size," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9514–9530, Sep. 2023.

[28] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 478–487, May/Jun. 2020.

[29] J. Ning, X. Huang, W. Susilo, K. Liang, X. Liu, and Y. Zhang, "Dual access control for cloud-based data storage and sharing," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1036–1048, Mar./Apr. 2022.

[30] Y. Guo, Z. Lu, H. Ge, and J. Li, "Revocable blockchain-aided attribute-based encryption with escrow-free in cloud storage," *IEEE Trans. Comput.*, vol. 72, no. 7, pp. 1901–1912, Jul. 2023.

[31] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 762–773, Apr.–Jun. 2022.

[32] S. Kim and I. Lee, "IoT device security based on proxy re-encryption," *J. Ambient* Intell. Humanized Comput., vol. 9, pp. 1267–1273, Aug. 2018.

[33] L. Xu, X. Wu, and X. Zhang, *CL-PRE: A Certificateless Proxy Re-Encryption Scheme for Secure Data Sharing With Public Cloud*. New York, NY, USA: Assoc. Comput. Mach., 2012. [Online]. Available: https://doi.org/10.1145/2414456.2414507

[34] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 2907–2919, Sep./Oct. 2022.

[35] M. Su, B. Zhou, A. Fu, Y. Yu, and G. Zhang, "PRTA: A proxy re-encryption based trusted authorization scheme for nodes on CloudIoT," *Inf. Sci.*, vol. 527, pp. 533–547, Jul. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519300660

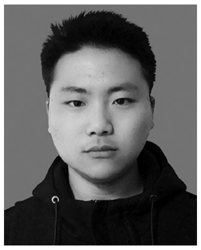
[36] K. Wang, Y. Yan, S. Guo, X. Wei, and S. Shao, "On-chain and off-chain collaborative management system based on consortium blockchain," in *Advances in Artificial Intelligence Security*, X. Sun, X. Zhang, Z. Xia, and E. Bertino, Eds. Cham, Switzerland: Springer Int., 2021, pp. 172–187.

[37] K. O.-B. O. Agyekum, Q. Xia, E. B. Sifah, C. N. A. Cobblah, H. Xia, and J. Gao, "A proxy re-encryption approach to secure data sharing in the Internet of Things based on blockchain," *IEEE Syst. J.*, vol. 16, no. 1, pp. 1685–1696, Mar. 2022.

[38] S.-J. Horng et al., "B-SPECS+: Batch verification for secure pseudonymous authentication in VANET," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 1860–1875, 2013.

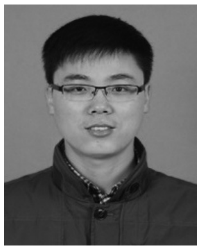
[39] "MIRACL core." Dec. 2023. [Online]. Available: https://github.com/miracl/core

[40] "Hyperledger." 2023. [Online]. Available: https://github.com/hyperledger/fabric



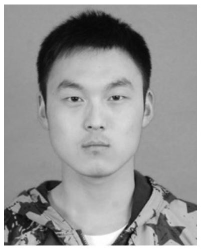
Fengqun Wang is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Anhui University, Hefei, China.

His research interests include IoT security, blockchain, and applied cryptography.



Jie Cui (Senior Member, IEEE) was born in Henan Province, China, in 1980. He received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2012.

He is currently a Professor and the Ph.D. Supervisor with the School of Computer Science and Technology, Anhui University, Hefei. He has over 150 scientific publications in reputable journals (e.g., IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON MULTIMEDIA), academic books and international conferences. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking.



Qingyang Zhang (Member, IEEE) was born in Anhui Province, China, in 1992. He received the B.Eng. degree and the Ph.D. degree in computer science from Anhui University, Hefei, China, in 2021.

He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University. His research interest includes edge computing, computer systems, and security.

Debiao He (Member, IEEE) received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2009.

He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. He has published over 100 research papers in refereed international journals and conferences, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and Usenix Security Symposium. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Prof. He is the recipient of the 2018 IEEE Systems Journal Best Paper Award and the 2019 IET Information Security Best Paper Award. He work has been cited more than 10 000 times at Google Scholar. He is on the editorial board of several international journals, such as *Journal of Information Security and Applications*, *Frontiers of Computer Science*, and *Human-centric Computing and Information Sciences*.

Chengjie Gu received the Ph.D. degree from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2012.

From 2012 to 2017, he was an Innovation Team Leader of the 38th Research Institute of CETC and conducted research and development in the communication and networking sector. He is currently a President of Security Research Institute in new H3C group. He is also studying for postdoctoral fellowship at USTC. He is a high-level Innovation Leader of Anhui province and a Cybersecurity Expert of Zhejiang province in China. His research interest includes network security and trusted network architecture.

Hong Zhong (Member, IEEE) was born in Anhui Province, China, in 1965. She received the Ph.D. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2005.

She is currently a Professor and the Ph.D. Supervisor with the School of Computer Science and Technology, Anhui University, Hefei. She has over 200 scientific publications in reputable journals (e.g., IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and IEEE TRANSACTIONS ON BIG DATA), academic books and international conferences. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking.