Hindawi
Security and Communication Networks
Volume 2022, Article ID 1317626, 11 pages
https://doi.org/10.1155/2022/1317626

Research Article

Security Enhancements for Data-Driven Systems: A Blockchain-Based Trustworthy Data Sharing Scheme

Yanping Wang,[1] **Xiaosong Zhang**,[1] **Xiaofen Wang**,[1] **Teng Hu**,[2] **Peng Lu**,[2] **and Mingyong Yin**[2]

[1]*UESTC, Institute for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China*
[2]*CAEP, Institute of Computer Application, China Academy of Engineering Physics, Mianyang, China*

Correspondence should be addressed to Teng Hu; mailhuteng@gmail.com

Received 9 August 2022; Accepted 14 September 2022; Published 11 October 2022

Academic Editor: Yuanyuan Huang

With the increasingly prominent value of big data, data sharing within enterprises and organizations has become increasingly popular, and many institutions have established data centers to achieve effective data storage and sharing. Meanwhile, cyberspace data security and privacy have become the most critical issue that people are concerned about since shared data often involves commercial secrets and sensitive information. At present, data encryption techniques have been applied to protect the security of the sensitive data stored in and shared by the data centers. However, the challenges of efficient data sharing, secure management of decryption keys, deduplication of the plaintext, and transparency and auditability of the data access arise. These challenges may obstruct the development of data sharing in data-driven systems. To meet these challenges, we propose a secure and trustworthy data sharing scheme and introduce blockchain, proxy re-encryption (PRE), and trusted execution environments (TEEs) into the data-driven systems. Our scheme mainly enables (1) automatic distribution and management of the decryption keys, (2) reduction of the reduplicative data, and (3) trustworthy data sharing and recording. Finally, we implement the proposed scheme and compare it with other existing schemes. It is demonstrated that our scheme reduces the computation and communication overhead.

1. Introduction

With the development of big data, the Internet of Things, and other network technologies, various kinds of data have been produced. The economic and social benefits of the data trigger the demand for sharing data between institutes and enterprises. Therefore, many organizations have established data centers utilizing the private or public cloud to realize effective data storage and sharing. Because data often involves business secrets and sensitive information, among others, data privacy and security are the key issues that people are concerned about, especially in large enterprises and scientific research institutes.

Encryption can be applied to protect the privacy and security of the sensitive data stored in and shared by the data center to a certain extent. Encrypted data sharing schemes [1, 2] are proposed, in which the data are encrypted by the owner and can only be decrypted by authorized users. In these scenarios [1, 2], an owner negotiates a session key with a group of users in advance so that they can share data with them. However, if a new user is added to the authorized sharing group, a new session key is needed to be negotiated, and data are required to be encrypted using this new session key. This inevitably introduces a large computation overhead if there are frequent changes in the sharing group.

In order to alleviate the above complex key management problems, the proxy re-encryption (PRE) techniques [3–5], which allow a proxy [e.g., cloud server (CS)] to convert a cipher of a delegator to different ciphers for different delegatees, have been used to share the data to different users dynamically without the complex key agreement and decrypt-then-encrypt operations. PRE properties make it a

practical approach to cloud-assisted data sharing. However, in order to avoid huge computation in PRE's cipher conversion, the CS may not generate the re-encryption ciphertext honestly [6]. Additionally, many PRE-based data sharing schemes [7, 8], cannot satisfy nonframeability. In other words, in these schemes, the CS may be maliciously framed for refusing to perform a re-encryption operation or for outputting a wrong reencrypted cipher when it indeed performs honestly.

Recently, blockchain has been applied in data sharing solutions [2, 9, 10], in which the encrypted data were stored in the off-chain data center (e.g., cloud), whereas the meta and data transfer log were recorded in the blockchain for data retrieval and auditing. Then, the data sharing schemes that combine the blockchain and PRE emerged such as in [11–14], where the encrypted data can be accessed by the authorized users with the help of a proxy server, and the misbehavior of the proxy server or the users in re-encryption can be hindered as all operations would be recorded in the blockchain and can be audited. Nevertheless, references [11–14] faced the efficiency challenge caused by the complex encryption/decryption and frequent interaction of data owners (DO) and users. On the one hand, frequent interactions and complex ciphertext transformations are heavy burdens to DO and CSs. On the other hand, the storage of large-size data is a great burden to the blockchain. In addition, in [11–14], the same data will be packaged into different ciphertexts, and the redundant plaintext copies would result in additional storage overhead. These challenges motivate us to propose a more efficient and versatile encrypted data sharing scheme.

In this paper, we combine blockchain, PRE, and trusted execution environments (TEEs) and propose a flexible and secure data sharing method for data-driven systems. By employing the PRE technology, our scheme allows the encrypted data to be transformed (by the CS) into different ciphertexts for different authorized users without the complex key agreement. Furthermore, by involving the blockchain, the misbehavior of the CS or the users in re-encryption can be recorded and audited. Meanwhile, the smart contract can automatically delegate the re-encryption key to authorized users so that the DO's computation and communication burden can also be reduced. Finally, by utilizing the TEEs, the smart contract can be executed in a secure enclave to protect the DO's private key. The major contributions of our scheme are as follows:

(1) Smart contract is employed to control the access of data, such that the decryption key's delegation can be executed automatically and the DO is not required to be online all the time, which greatly reduces the computation and communication burden of the DO and makes the data sharing convenient.

(2) Our scheme utilizes the tamper-proof and consensus properties of the blockchain, and the transfer logs of data requests and replays are recorded in the distributed ledger, which realizes the trustful recording and the real-time monitoring.

(3) In our scheme, the duplicate data can be quickly detected to avoid redundancy, and its storage request will be refused. Therefore, the ciphertexts corresponding to the same plaintext can be reduced.

(4) We conduct experiments to evaluate the performance of our proposed scheme, and the results show that our scheme is more efficient than the existing schemes [11–14] with respect to computation overhead and cipher size.

The rest of this paper is organized as follows: Section 2 surveys the related works. The preliminaries are presented in Section 3. The system model and design goals are described in Section 4. Section 5 introduces the detailed proposal, including the data release and retrieval. The analysis and simulation of the proposed solution are shown in Section 6. Finally, in Section 7, we draw our conclusion.

2. Related Works

With the rapid development of blockchain technologies, the schemes [2, 9, 15, 16] of decoupling the storage layer and the blockchain have been proposed to achieve efficient and reliable data sharing, especially in large-scale data-driven systems. In these cases, the data generated from the source are stored in the off-chain data center, and the meta (e.g., digest) is recorded on the blockchain for efficient data retrieval. When a user queries data from the blockchain, the distributed ledger's retrieval mechanism can help users quickly retrieve the queried information from the blockchain, which greatly improves the efficiency and credibility of the system. However, most of these schemes use blockchain just as a distributed and immutable database, and the issues such as trusted access control have not been completely solved.

The concept of PRE was initially introduced and constructed in [3], in which the DO controls the delegation of data access with the help of a CS. Based on this concept, Ran and Susan [4] proposed a secure PRE scheme against chosen ciphertext attacks. Next, Weng et al. [5] proposed a conditional PRE, which achieves a more fine-grained delegation. In order to ensure secure and efficient data sharing, PRE technology is used in [11–14] to realize multisharing controls of ciphertext for blockchain-based big data storage. In schemes [11–14], DO outsource their encrypted data to the cloud using identity-based encryption and grant legitimate users access to the data. However, these schemes face heavy communication and computation costs. Additionally, schemes [11–14] either rely on a proxy to fully manage their data or require the DO to always be online to delegate the decryption key to the data user, which means either the data transparency and auditability cannot be achieved or the DO needs to be online all the time.

For achieving privacy-preserving and automatic data sharing, Li et al. [17] proposed a blockchain-based privacy-preserving data sharing scheme with rewards, which uses smart contracts to automatically generate the decryption keys for users, and the TEEs are used to ensure the security of secret keys in smart contracts. Wang et al. [18] proposed a

TEEs-based smart contract execution scheme, which is used to share private data with fine-grained access control for smart grids. Lei et al. [19] proposed a multiparty data sharing platform that combines blockchain and TEEs and realizes automated data sharing. However, these schemes face communication and computation burdens caused by fine-grained access control.

Based on the above research, we provide a trust and efficient data sharing scheme. We incorporate blockchain, TEEs, and proxy re-encryption to achieve secure data sharing while achieving (1) efficient one-to-many sharing of data, (2) automatic distribution and management of the decryption keys, (3) reduced reduplicative data storage, and (4) trustworthy data transmission and records.

3. Preliminaries

This section briefly outlines the preliminaries about pairing groups, blockchain technologies, and PRE. The key notations involved in this paper are summarized in Table 1.

3.1. Pairing Groups. Let $pp = (p, \mathbb{G}, \mathbb{G}_T, e)$ be the pairing parameter, where $\mathbb{G}, \mathbb{G}_T$ is the finite groups of order p and e an efficiently computable bilinear map from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$, which satisfies the following:

(1) Bilinearity: for any generator $P, Q \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, the equation $e(aP, bQ) = e(P, Q)^{ab}$ holds;

(2) Nondegenerability: $e(P, Q) \neq 1$;

(3) Computability: e can be efficiently computed.

Difficult problems based on the above bilinear pairings are defined as follows:

Definition 1. (DL assumption) [3]. Let $\mathbb{G}$ be a cyclic group. The Discrete Logarithm (DL) assumption is that, for all $P \in \mathbb{G}$ and $a \in \mathbb{Z}_p$, given an input $\{P, aP\}$, the probability of outputting a is negligible for any polynomial time algorithm.

Definition 2. (3-QBDH assumption) [5]. Let $pp = (p, \mathbb{G}, \mathbb{G}_T, e)$ be the pairing parameter and P a generator of $\mathbb{G}$. The 3-quotient bilinear Diffie–Hellman (DBDH) assumption on pp is as follows: for any unknown $a, b \in \mathbb{Z}_p$, given $\{P, -aP, aP, a^2P, bP\}$, the probability of computing $e(P, P)^{b/a^2}$ is negligible for any polynomial time algorithm.

3.2. Some Basic Knowledge of Blockchain. With the launch of the bitcoin network [20], the concept of blockchain has become widely known to the public. As a decentralized distributed ledger maintained by multiple parties, the primary purpose of blockchain is to solve the trust problem in untrustworthy distributed environments by using peer-to-peer (P2P) network schemes, consensus algorithms, asymmetric encryption, password hashing, and other technologies. The blockchain can be used as a secure data management system [15] to ensure data integrity and availability. It can also be used as a supervision and audit platform [21] to achieve transparent supervision of the data.

Table 1: Key notions.

Notions	Description
$\|$	Data concatenation
h, h_1, H	Cryptographic hash functions
$(sk_{\text{tee}}, pk_{\text{tee}})$	Private/public key pair of a TEE
(sk_o, pk_o)	Private/public key pair of DO
(sk_c, pk_c)	Private/public key pair of DC
(sk_s, pk_s)	Private/public key pair of CS
(dk, ek)	Randomly generated key pair by DO
$SEnc(\cdot), S\ De\ c(\cdot)$	Symmetric encryption and decryption
$AEnc(\cdot), A\ De\ c(\cdot)$	Asymmetric encryption and decryption
$sig_{sk_i}(\cdot)$	Signature under the secret key sk_i
k_{st}	State key of smart contracts
$k_{o \longrightarrow c}$	Reencryption key for DC
CF, CF'	Original ciphertext and reencrypted ciphertext

Additionally, the blockchain can be used as a platform [17] that achieves secure and trusted data processing by utilizing smart contracts (self-executing programs with clauses clearly specified by the underlying code and deployed on the blockchain).

3.3. Trusted Execution Environments. As data in smart contracts are transparent on the blockchain, users' private information can easily be exposed [17]. TEEs, such as Intel Software Guard Extensions (SGX) [22], TrustZone [23], and MultiZone [24], can be utilized to solve this problem. The TEEs enable secure execution of programs on untrusted hosts (e.g., cloud), as the programs can be run in a protected manner by isolating all the operations against the outside world. They also allow remote verifiers to ascertain a device's current configuration and behavior *via* remote attestation. It is worth noting that, *via* remote attestation, a TEE can build a secure channel for the user and other TEEs to communicate with it securely. These properties make TEEs a good choice for processing and sharing private data. For example, Bowman et al. [25] proposed a TEEs-based private data process scheme, named private data objects (PDOs), which allows mutually untrusted parties to work on private data based on preagreed policies, and the open-source code is provided in [26].

3.4. Proxy Re-encryption. The concept of the RRE was introduced by Blaze et al. [3], in which a semitrusted proxy server is delegated to convert a delegator's ciphertext to a delegatee's without the leakage of the corresponding plaintext. The PRE can be used for secure data sharing in cloud environments, which usually consists of the following five algorithms:

$\mathbf{KeyGen}(\lambda) \longrightarrow (\mathbf{pk}, \mathbf{sk})$: on the input of the security parameter λ, this algorithm outputs a public/private key pair (pk, sk);

$\mathbf{Re-Key}(\mathbf{sk_o}, \mathbf{pk_c}) \longrightarrow \mathbf{r_{o \longrightarrow c}}$: on the input of a user's private key sk_o and another user's public key pk_c, this algorithm outputs a re-encryption key $r_{o \longrightarrow c}$;

Encryption $(\mathbf{pk_o}, \mathbf{M}) \longrightarrow \mathbf{CF}$: on the input of a user's public key pk_o and the plaintext M, this algorithm outputs a ciphertext CF under the public key pk_o;

Re – Encryption $(\mathbf{CF}, \mathbf{r_{o\longrightarrow c}}) \longrightarrow \mathbf{CF'}$: on the input of the ciphertext CF under the public key pk_o and the reencryption key $r_{o\longrightarrow c}$, this algorithm outputs a ciphertext CF' under the public key pk_c;

De cryption $(\mathbf{sk_c}, \mathbf{CF'}) \longrightarrow \mathbf{M}$: on the input of a user's secret key sk_c and the reencrypted ciphertext $CF\prime$, this algorithm outputs a plaintext M.

4. System Model

In this section, we illustrate the framework, outline the threat model, and design goals of the proposed scheme.

4.1. Framework. The schematic diagram of our framework is shown in Figure 1, in which the consensus is separated from the execution of the smart contract. Similar to Ekiden [27], our framework consists of a CS, consensus nodes, participant nodes, and authorities. Each component and its role are described below.

4.1.1. Cloud Server. It is usually a data center responsible for storing and securely sharing the encrypted data for the users with the help of a TEE (this TEE in the cloud is called sTEE). The CS loads the smart contract, executes it in the sTEE to generate a re-encryption key, and performs the reencrypting operations.

4.1.2. Consensus Nodes. There are two types of consensus nodes: without TEEs and with TEEs. A consensus node without TEEs is responsible for maintaining the blockchain ledger and realizing basic blockchain functions such as packaging blocks and verifying blocks. Besides maintaining the ledger, consensus nodes with TEEs play the role of key management committees (these TEEs are called kTEEs) and are responsible for managing secret keys and remotely attesting to the sTEE.

4.1.3. Participating Nodes. They are the blockchain users, including the DO and data consumers (DC). DO is responsible for data release; DC requests the data by revoking the smart contract. After that, DC obtains the reencrypted data, which can be decrypted using their private key.

4.1.4. Authorities. There are two types of authorities, certificate authority (CA) and judgment authority (JA). The CA is responsible for membership enrollment and certificate distribution, and the JA is responsible for judging whether malicious behaviors have been performed.

Algorithm 1 further illustrates the interactions of Figure 1 among CS, consensus nodes, and participating nodes. Algorithm 1 shows that the interactions include two parts, data release and data retrieval, and DC can obtain data M when the algorithm ends.

4.2. Threat Model and Design Goals. In our system, the authorities are trusted, and DO will honestly share the ciphertext of the data. CS and some unauthorized DCs are curious about DO's data, and CS may not honestly transfer the cipher to DC. Moreover, the openness of blockchain enables the analysis of the transaction information (such as input and output) [28], which may cause the leakage of DO's data or DC's identity and attributes. We further assume the data and program can be securely stored and executed in the TEEs.

Based on the above security hypothesis, our goals are achieved if the below properties are satisfied.

4.2.1. Data Confidentiality. The DO's data should be kept confidential to CS during the storage and computation. Moreover, except for the DO and the authorized DC, other users cannot obtain the original data.

4.2.2. Nonredundant Storage. The storage requests of duplicate data should be quickly detected and refused for reducing storage costs.

4.2.3. Verifiable Integrity. When obtaining the data from CS, data integrity can be easily verified by DC.

4.2.4. Anonymity. The DC's identity and attributes should not be recognized by anyone during the data requesting process.

4.2.5. Transparency and Auditability. DO should know whom their data are shared with. Besides, the access and computation processes should be auditable.

4.2.6. Efficient Computation. The data encryption/decryption should avoid the heavy cryptographic overhead and save computation costs as much as possible.

5. The Proposed Approach

The proposed data sharing scheme includes three phases: system initialization, data release, and data retrieval. CA generates the system parameters, and each entity generates a private/public key pair and then registers to CA in the system phase. In the data release phase, DO releases their encrypted data and delegates the corresponding secret key shares to a group of kTEEs. Finally, in the data retrieval phase, DC invokes the smart contract and will obtain a reencryption ciphertext, and then DC decrypts the re-encryption ciphertext to recover DO's data.

5.1. System Initialization. CA chooses a security parameter λ and generates the pairing parameter $pp = (p, \mathbb{G}, \mathbb{G}_T, e)$. CA also chooses a generator P in $\mathbb{G}$, a symmetric encryption algorithm *SEnc* (such as AES), an asymmetric encryption algorithm *AEnc* (such as ElGamal), and three secure cryptographic hash functions, h, h_1, and H, where h:

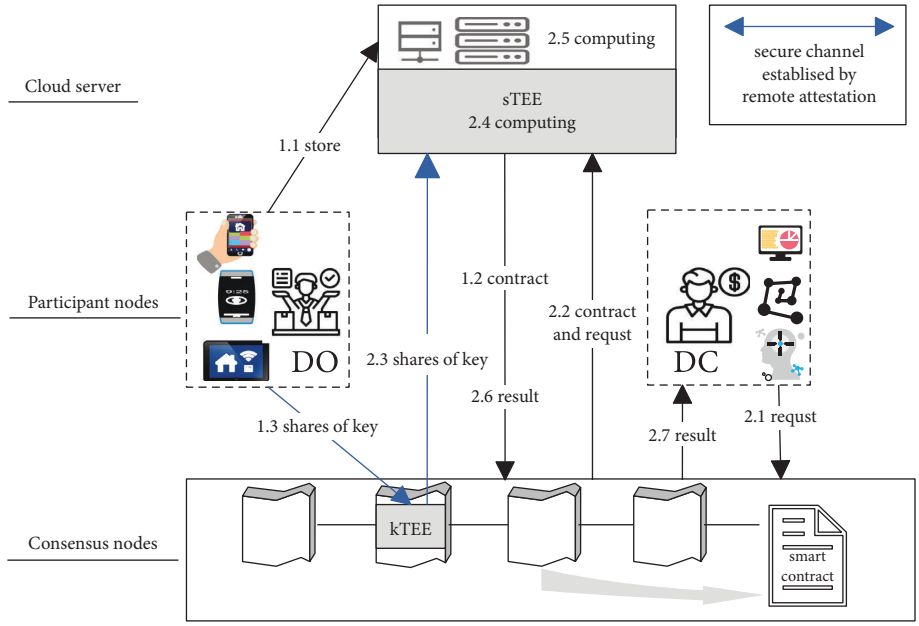

Figure 1: The proposed framework.

Require: DO: the encrypted data CF, the contract's program code Contract, the secret key shares $dk_1, dk_2, \ldots, dk_n$; DC: the request req.
Ensure: the data M.
procedure DATA RELEASE:
Step 1.1: DO sends CF and Contract to CS;
Step 1.2: CS publishes Contract to blockchain;
Step 1.3: DO checks Contract
if the Contract in the blockchain is correct, **then**
DO sends the secret key shares dk_i to the kTEE i.
procedure **Data retrieval:**
Step 2.1: DC revokes the smart contract with input req;
Step 2.2: CS loads Contract and req into the sTEE;
Step 2.3: sTEE performs remote attestation with kTEEs
if sTEE environment and loaded data are correct, **then**
The kTEE i transmits dk_i to the sTEE;
Step 2.4: CS executes the smart contract in the internal sTEE to obtain a reencryption key;
Step 2.5: CS computes the reencrypted ciphertext $CF\prime$ outside the sTEE;
Step2. 6: CS sends the reencrypted data $CF\prime$ to the blockchain;
Step 2.7: DC obtains $CF\prime$ and decrypts it to obtain M.

Algorithm 1: The process of the proposed scheme.

$\{0,1\}^{l_1} \leftarrow \{0,1\}^*$, $h_1: \mathbb{Z}_p \leftarrow \{0,1\}^*$, and $H: \{0,1\}^{l_2} \leftarrow \mathbb{G}_T$, and l_1 and l_2 represent the output lengths of hash. CA stores the system parameters $(pp, P, SEnc, AEnc, h, h_1, H)$ on the blockchain.

Each participant node i picks a private key $sk_i \in \mathbb{Z}_p$, computes the public key $pk_i = sk_iP$, and then registers to CA. CA then issues a certificate cert_i to user i. The certificate is combined with the user's public key pk_i and attributes.

The CS picks a private key $sk_s \in \mathbb{Z}_p$, computes the public key $pk_s = sk_sP$, and then registers to CA. CS and key management nodes initialize their TEEs and send the necessary information (e.g., the TEE's public key pk_{tee}) to the blockchain.

5.2. Data Release. In our scheme, data are encrypted and stored off-chain while the related information is stored on-chain, and the DO can specify whom their data can be shared with. The data release (Figure 2) is conducted as follows.

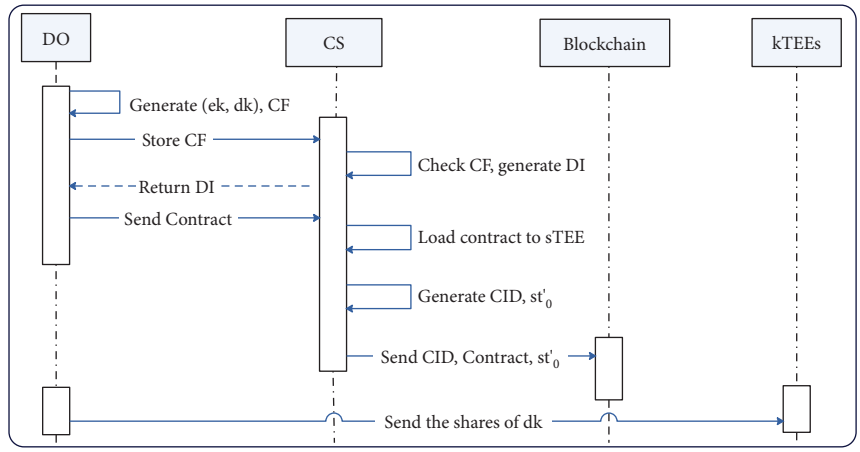

Figure 2: Sequence diagram of data release.

5.2.1. Data Encryption and Storage. DO randomly generates a private/public key pair (dk, ek) that satisfies $ek = dk \cdot P$. Then, DO randomly chooses $r \in \mathbb{Z}_p$ and computes $s = h(m)$, $C_0 = SEnc_s(m)$, $C_1 = r \cdot ek$, and $C_2 = s \oplus H(e(P,P)^r)$, where $SEnc_s$ represents the symmetric encryption under the secret key s. DO sends $CF = (C_0, C_1, C_2)$ to CS. If C_0 is not duplicated with other ciphers (this means m is different from other data), DO will receive a data retrieval index DI and a timestamp TS from the CS, where $DI = sig_{sk_s}(h(CF) \| \text{TS})$. DO checks the validity of the DI using the CS's public key pk_s. If it is valid, then it continues; otherwise, it aborts.

5.2.2. Smart Contract Creation. The smart contract is responsible for generating the re-encryption key for the authorized DC, and the contract's creation is carried out as follows:

(1) DO creates a smart contract Contract, which is written in the form of program codes. Then, DO chooses a state key k_{st} and generates k'_{st} by encrypting k_{st} under the sTEE's public key pk_{tee}. DO sends the Contract to the CS. The Contract contains the related information $RI = \{ek, A, W, DI, k_{st}, TS, sig_{sk_o}(ek, A, W, DI, k'_{st}, TS)\}$, where A is the access attributes, W is the keywords set, k'_{st} is the encrypted state key, and $\text{sig}_{sk_o}(\cdot)$ is the signature under DO's private key sk_o.

(2) CS loads the code of Contract into the sTEE, and then the sTEE generates a new contract ID, namely, CID, and decrypts k'_{st} using its secret key sk_{tee} to recover the state key k_{st}. Then, the sTEE encrypts the initial contract state as $st'_0 = SEnc_{k_{st}}(\overrightarrow{0})$ and outputs $\{\text{CID}, \text{Contract}, st'_0, \pi\}$, where π is a correctness proof generated using sTEE's secret key sk_{tee}. After that, CS sends the output to the blockchain. The consensus nodes will verify π, pack the legitimate $\{\text{CID}, \text{Contract}, st'_0, \pi\}$ into a block, and record it on the blockchain.

(3) After the Contract has been confirmed in the blockchain, DO sends CID and the shares of dk to the kTEEs. The dk is shared using the secret-sharing schemes [29, 30], and each share is encrypted using the corresponding kTEE's public key. The security feature of TEEs ensures that dk is kept secret against other nodes.

5.3. Data Retrieval. The anonymous data retrieval is conducted as shown in Figure 3, which is comprised of three phases: off-chain re-encryption key generation, cipher re-encryption, and data decryption. In the first phase, DC requests the cipher by invoking the smart contract, and then the smart contract executed in the sTEE generates a re-encryption key for the authorized DC. In the second phase, CS reencrypts the cipher and sends it to DC. In the last phase, DC receives the encrypted data and decrypts it to obtain the plaintext.

5.3.1. Off-Chain Re-encryption Key Generation. DC retrieves the interested keyword W_i on the blockchain and obtains CID from the related information RI. After checking the contents of the contract, DC invokes the smart contract with input $\text{req} = \{\text{CID}, AEnc_{ek}(DI, pk_c, \text{cert}_c)\}$, where $AEnc_{ek}$ represents the asymmetric encryption under the public key ek. Then, the CS loads the corresponding contract state st'_{old} and req into sTEE, and the sTEE performs remote attestation with kTEEs to attest the sTEE environment. The loaded smart contract and data are correct. After passing the attestation, the shares of the decryption key dk will be transmitted to sTEE through secure channels. Once obtaining enough shares, the smart contract in sTEE performs the following steps:

(1) recovers the decryption key dk and state key k_{st};

(2) ecrypts st'_{old} using the state key k_{st} and obtains the old state st_{old}.

(3) ecrypts *req* using the decryption key dk and obtains the certificate of DC.

(4) hecks whether DC satisfies the access condition by verifying DC's attributes in their certificate. If it is satisfied, it continues. Otherwise, it jumps to step 7.

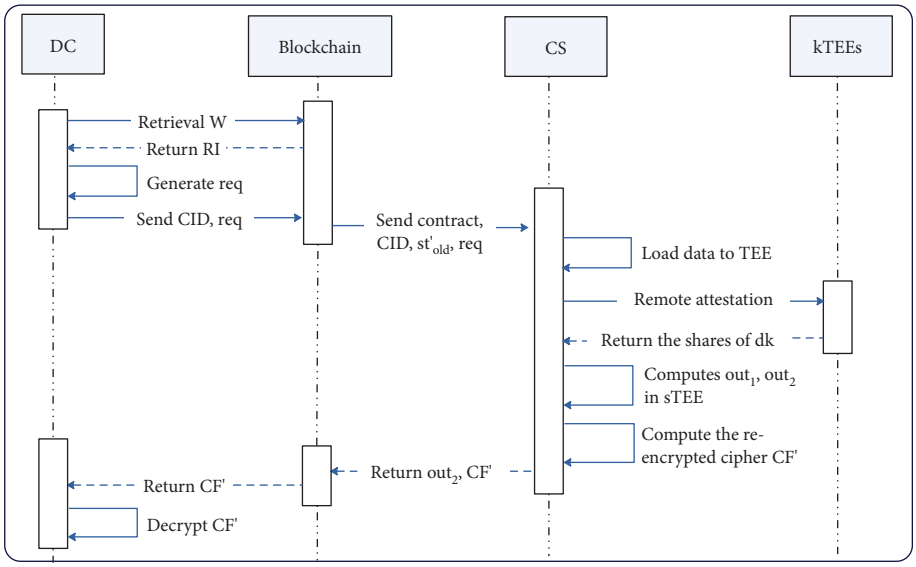

Figure 3: Sequence diagram of data retrieval.

(5) omputes $k = h_1(DI \| pk_s \| ek \| sk_{\text{tee}} \cdot pk_c)$.

(6) omputes a re-encryption key $k_{o\longrightarrow c} = (-\text{dk}) \cdot k \cdot pk_c$.

(7) pdates the contract state as st_{new} and computes $st'_{\text{new}} = SEnc_{k_{st}}(st_{\text{new}})$.

(8) utputs the executed results.

When the execution ends, all involved keys and intermediate results of the off-chain smart contract execution in sTEE can be securely erased [18]. There are two outputs: $\text{out}_1 = \{DI, k_{o\longrightarrow c}\}$ and $\text{out}_2 = \{st'_{\text{new}}, h(k_{o\longrightarrow c}), \pi\}$ (if DC is illegal, then $\text{out}_1 = \perp$ and $\text{out}_2 = \{st'_{\text{new}}, \perp, \pi\}$).

5.3.2. Cipher Re-encryption. CS retrieves the cipher CF according to DI and reencrypts CF to obtain $CF\prime = \{C'_0 = C_0, C'_1 = e(C_1, k_{o\longrightarrow c}), C'_2 = C_2\}$. CS then sends the reencrypted cipher to a temporary location, and a transaction $\text{tran} = \{\text{out}_2, \text{url}, h(CF\prime), \text{sig}_{sk_s}(\text{out}_2, \text{url}, h(CF\prime))\}$ is sent to the blockchain by CS, where url is the link of the temporary location that $CF\prime$ stores. For the transaction tran, consensus nodes check the validity of out_2 through the proof π provided by sTEE, check the validity of the url and $h(CF\prime)$ through the signature provided by CS, and maintain the consistency of state through consensus schemes.

5.3.3. Data Decryption. DC can retrieve the location url from the blockchain and download the cipher. After that, DC computes $k = h_1(DI \| pk_s \| ek \| sk_c \cdot pk_{\text{tee}})$. DC then decrypts the cipher $CF\prime$ by computing $s = C'_2 \oplus H(C'_1(-k) \cdot (-sk_c))$ and $m = S\ De\ c_s(C'_0)$, where $S\ De\ c_k$ is the symmetric decryption under the key s. DC verifies the data by checking if $h(m) \stackrel{?}{=} s$. If yes, DC accepts it; otherwise, DC rejects it and complains to the authority JA.

5.4. Claim. JA firstly requests the CS to provide the cipher CF and the sTEE's output out_1. After confirming the correctness of CF (using the index DI in the blockchain), JA computes the hash of the re-encryption key $H(k_{o\longrightarrow c})$ and compares if it is equal to that in out_2 of the blockchain. If yes, JA computes the re-encryption cipher $(\overline{C_0}, \overline{C_1}, \overline{C_2})$ and compares if $\overline{C_0} = C'_0$, $\overline{C_1} = C'_1$, and $\overline{C_2} = C'_2$ hold. If they hold, it ignores this complaint; otherwise, the CS has misbehaved, JA takes action accordingly.

6. Analysis and Evaluation

In this section, we analyze the security properties and evaluate the performance of the proposed scheme.

6.1. Security Analysis. Our scheme achieves the security properties of correctness, confidentiality, verifiable integrity, transparency, and auditability.

Theorem 1. *If DO, DC, and CS execute the scheme honestly, then DC can obtain DO's data correctly.*

We can prove Theorem 1 by verifying the following equation:

$$\begin{aligned} s &= C'_2 \oplus H(C'_1(-k) \cdot (-sk_c)) \\ &= s \oplus H(e(P,P)^r) \oplus H\left(e(C_1, k_{o\longrightarrow c})^{(-k)\cdot(-sk_c)}\right) \\ &= s \oplus H(e(P,P)^r) \oplus H\left(e(r \cdot dk \cdot P, (-dk) \cdot k \cdot pk_c)^{(-k)\cdot(-sk_c)}\right) \\ &= s \oplus H(e(P,P)^r) \oplus H(e(P,P)^r) = s. \end{aligned} \tag{1}$$

Theorem 2. *Our scheme achieves confidentiality if the 3-QBDH assumption holds.*

We prove the confidentiality of our scheme by proving that secret s cannot be recovered from the ciphertext by unauthorized users. We constructed an algorithm $\mathcal{B}$ that is given the pairing parameters $(\mathbb{G}, \mathbb{G}_T, p, e)$ and an instance $(P, A_0 = -aP, A_1 = aP, A_2 = (a^2)P, B = bP, T)$ and aimed to decide whether $T = e(P,P)^{b/a^2}$. $\mathcal{B}$ controls a hash oracle and runs an algorithm $\mathcal{A}$ (aimed to break the confidentiality of s) as a subroutine. We can prove that if $\mathcal{A}$ breaks the confidentiality of s, then $\mathcal{B}$ can break the 3-QBDH problem.

Before starting, we define two lists, $\mathcal{L}_h$ and $\mathcal{L}_c$, where $\mathcal{L}_h$ is the list of honest users and $\mathcal{L}_c$ is the list of corrupt users.

(i) Init phase: $\mathcal{A}$ prepares lists $\mathcal{L}_h$ and $\mathcal{L}_c$ and outputs $i^* \in \mathcal{L}_h$ as the challenger user. Let sk_{i^*}, sk_i be the random numbers chosen from $\mathbb{Z}_p$. The public key for the challenge user i^* is set as $pk_{i^*} = sk_{i^*}A_2$ and the corresponding secret key is $a^2 sk_{i^*}$. Public keys of other honest users $i \in \mathcal{L}_h$ are set as $pk_i = sk_i A_1$, and the corresponding secret key is $a \cdot sk_i$. Public keys of corrupt users $i \in \mathcal{L}_c$ are $pk_i = sk_i P$, and the corresponding secret key is sk_i. It should be noted that the corrupt users' key pair $(sk_i, pk_i)_{i \in \mathcal{L}_c}$ is known as $\mathcal{A}$.

(ii) Find phase: $\mathcal{A}$ plays the role of user j and queries a re-encryption key of user i from $\mathcal{B}$. If $i = i^*$ and $j \in \mathcal{L}_h$, $\mathcal{B}$ randomly chooses $k \in \mathbb{Z}_p$ and computes $k_{i \longrightarrow j} = sk_j \cdot k \cdot (-sk_{i^*}) \cdot A_0 = -(sk_{i^*}a^2) \cdot (sk_j a) \cdot k \cdot P$. If $i, j \neq i^*$, $i \in \mathcal{L}_h$, and $j \in \mathcal{L}_h$, $\mathcal{B}$ randomly chooses $k \in \mathbb{Z}_p$ and computes $k_{i \longrightarrow j} = sk_j \cdot k \cdot (-sk_i) \cdot P = -(sk_i a) \cdot (sk_j a) \cdot k \cdot P$. If $i \in \mathcal{L}_h$, $i \neq i^*$ and $j = i^*$, $\mathcal{B}$ randomly chooses $k \in \mathbb{Z}_p$ and computes the rekey $k_{i \longrightarrow j} = sk_{i^*} \cdot k \cdot (-sk_j) \cdot A_1 = -(sk_j a) \cdot (sk_{i^*}a^2) \cdot k \cdot P$. If $i \neq i^*$, $i \in \mathcal{L}_h$, and $j \in \mathcal{L}_c$, $\mathcal{B}$ randomly chooses $k \in \mathbb{Z}_p$ and computes $k_{i \longrightarrow j} = sk_j \cdot k \cdot (-sk_i) \cdot A_0 = -(sk_i a) \cdot sk_j \cdot k \cdot P$. If $i \in \mathcal{L}_c$, $\mathcal{B}$ computes $k_{i \longrightarrow j} = (-sk_i) \cdot k \cdot pk_j$.

(iii) Challenge phase: $\mathcal{A}$ chooses two numbers (s_0, s_1) and sends them to $\mathcal{B}$. $\mathcal{B}$ chooses s_b, where $b \in \{0,1\}$. $\mathcal{B}$ sets the challenge cipher C^* as $C_1^* = sk_{i^*}B$ and $C_2^* = s_b \oplus H(T)$ and returns C^* to $\mathcal{A}$.

(iv) Guess phase: $\mathcal{A}$ outputs its decision of $s_{b\prime \in \{0,1\}}$. If $b\prime = b$, $\mathcal{B}$ outputs 1, which means $T = e(P,P)^{b/a^2}$ and outputs 0 otherwise, which means T is a random number of $\mathbb{G}_T$.

The confidentiality of secret s is converted to the hardness of the 3-QBDH problem. If T is a random number, then the probability of $\mathcal{A}$ to break the confidentiality of our scheme is 1/2. If $T = e(P,P)^{b/a^2}$, then C^* is a valid ciphertext of m_b with $r = b/a^2$, $pk_{i^*} = sk_{i^*}A_2 = sk_{i^*}a^2 P$, and $B = bP$. Therefore, if $\mathcal{A}$ can break the confidentiality of our scheme with advantage ε, then $\mathcal{B}$ can break the 3-QBDH assumption with advantage $1/2\varepsilon$.

Theorem 3. *Our scheme reduced the redundant storage, meanwhile, satisfied the verifiable integrity.*

In the data release phase, data m are encrypted under key s, equal to $h(m)$. Therefore, the same data m will be encrypted under the same symmetric key s, and the ciphertext $C_0 = SEnc_s(m)$ will be the same. Therefore, CS can quickly detect whether plaintext m has already existed in the database and refuse the redundant data's storage request. Furthermore, when DC decrypts key s from ciphertexts C_1', C_2' and then recovers data m' from C_0 using s, they can compare if $h(m\prime)$ equals s to verify the integrity. If the verification fails, DC can infer that the ciphertext has been modified or has not been generated correctly.

Theorem 4. *Our scheme satisfies the anonymity for DC.*

In the data retrieval phase, DC uses the regenerated pseudonym addresses to request the cipher. Meanwhile, the certificate is encrypted and can only be decrypted in the sTEE. Therefore, others cannot know the real identity and attributes of DC, and no one can recognize DC from the pseudonym. Therefore, our scheme achieves anonymous data retrieval.

Theorem 5. *Our scheme achieves transparency and auditability.*

The data access events are transparently recorded in the unforgeable ledger as transactions, publicly auditable to DO and JA. If the CS did not honestly generate the reencrypted cipher or if the CS was maliciously accused of returning the wrong reencrypted ciphertext, JA could easily detect it using the information in the ledger.

6.2. Performance Evaluation. We evaluate the computation and communication overheads of our scheme and then compare them with the related blockchain-based PRE schemes [11–14]. The symmetric pairings e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ over the elliptic curve E: $y^2 = x^3 + 3x \bmod p$ with embedding degree 2 are constructed, the field size is 520 bits, and the group order is 160 bits. The bilinear pairing achieves the security level of 80 bits. Our simulations are supported by the MIRACL library [31], and our execution environment is performed on a laptop with AMD Ryzen 5 3550H 2.10 GHz processor and 16.00 GB memory.

6.2.1. Computation Overhead. The key cryptographic operations are Par, Exp, and Sm, which means the bilinear pairing operation, the exponentiation operation in $\mathbb{G}_T$, and the scalar multiplication operation in $\mathbb{G}$, respectively. Based on this setting, the main computational costs of our scheme are listed in Table 2. In the data release phase, DO performs 1 Sm operation and 1 Par. In the data retrieval phase, the smart contract performs 2 Sm operations, the CS performs 1 Par operation, and DC performs 1 Exp operation.

In order to demonstrate the efficiency of our scheme, we compare our scheme with the related schemes [11–14] in

Table 2: The computational complexity of each phase.

Phase	Entity	Operation
Data release	DO	Sm + Exp
Data retrieval	Smart contract	2 Sm
	CS	Par
	DC	Exp

Table 3: Computational comparison.

	Data release	Data retrieval
[11]	3 Sm + Exp	4 Sm + 2 Par
[12]	2 Sm + 2 Exp	8 Sm + 2 Par
[13]	2 Sm + Par	Sm + 2 Par + Hp
[14]	3 Sm + Par	3 Sm + 4 Par
Ours	Sm + Exp	Par + Exp

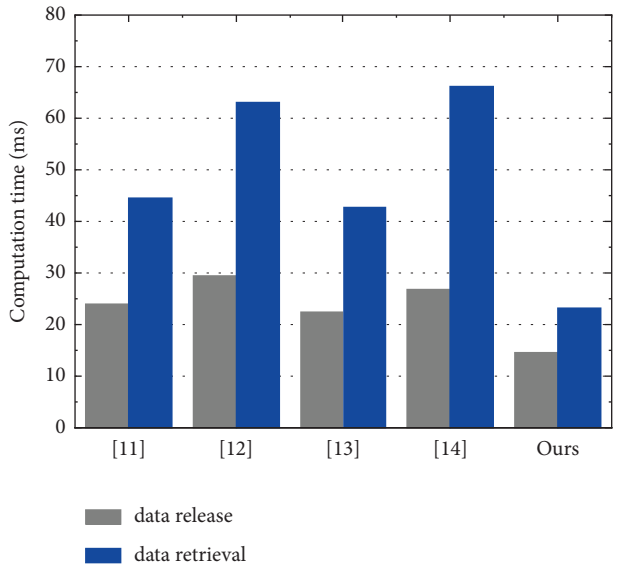

Figure 4: Computational overhead comparison.

Table 4: Communication overhead comparison.

	Ciphertext	Reencrypted ciphertext
[11]	$1\|G\| + 1\|G_T\|$	$4\|G\| + 1\|G_T\|$
[12]	$2\|G\| + 2\|G_T\|$	$1\|G\| + 2\|G_T\|$
[13]	$2\|G\|$	$3\|G\| + 1\|G_T\|$
[14]	$3\|G\| + 1\|G_T\|$	$2\|G\| + 2\|G_T\|$
Ours	$2\|G\|$	$1\|G\| + 1\|G_T\|$

terms of the above operations. Reference [11] proposed a blockchain-based data trade scheme. Reference [12] proposed a data sharing scheme for the scenario of multiple DCs, in which the PRE and smart contracts were integrated to achieve the privacy-preserving share of medical data. References [13, 14] proposed identity-based PRE approaches to achieve secure data sharing for cloud-assisted systems. We compare our scheme with these schemes [11–14] as we all

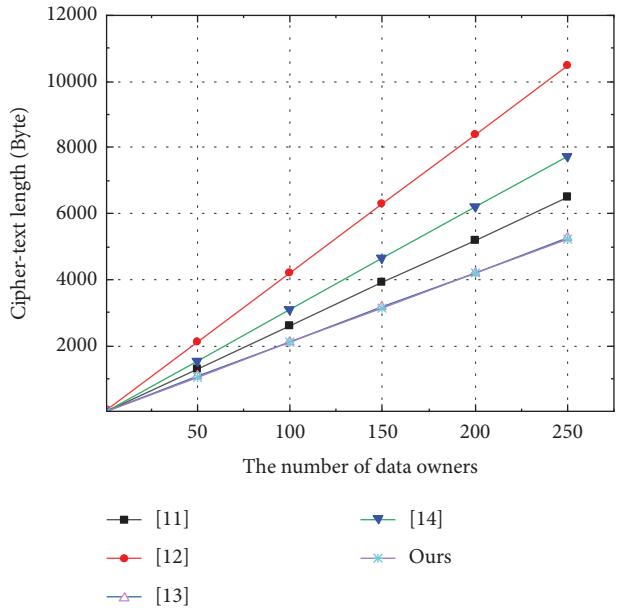

Figure 5: Communication overhead of ciphertext.

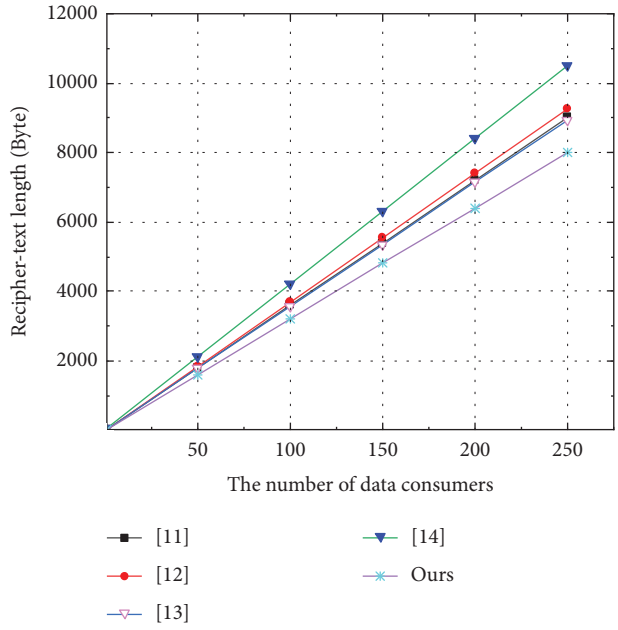

Figure 6: Communication overhead of the reencrypted ciphertext.

utilized the PRE and blockchain and achieved secure data sharing.

Table 3 shows the comparison of operations Par, Exp, and Sm among our scheme and schemes [11–14]. We can see that our scheme needs the fewest Par, Exp, and Sm operations. Figure 4 shows the total time of these schemes, from which we can see that the computation time of data release in our protocol is 14.7 ms, which is much smaller than 23.9, 29.4, 22.3, and 22.9 ms of [11–14]. The computation time of data retrieval in our protocol is 23.2 ms, which is also smaller than 44.6, 63, 42.8, and 66.2 ms of [11–14].

6.2.2. Communication Overhead. To evaluate the communication overhead, we denote the sizes of a scalar value in $\mathbb{Z}_p$, the group elements in $\mathbb{G}$, and in $\mathbb{G}_T$ by $|Z_p|, |G|$, and $|G_T|$, respectively. We choose SHA-256 as the hash function of h. The symmetric encryption algorithm is AES-256. The signature used to sign a transaction of blockchain is ecdsa-with-SHA256. Based on these, Table 4 compares the communication costs in [11–14] in terms of encryption overhead and re-encryption overhead. Figure 5 and Figure 6 show the comparison results, from which we can see that the ciphertext length of our protocol and that of [13] are identical, which are shorter than those in [11, 12, 14], and the length of the re-encryption ciphertext of our scheme is shorter than those in [11–14]. Considering that our scheme has a significant advantage in computational time, our scheme is more efficient in the aspect of both computational overhead and communication overhead.

7. Conclusion

This paper proposes a flexible and secure data sharing method for data-driven systems. In order to ensure confidentiality and reliability, data are encrypted and then stored off-chain. In contrast, the relevant information, such as digest, is stored on-chain, and data can be efficiently shared with authorized users with the help of a CS. The smart contract is employed to control data access such that the key delegation can be automatically executed and the DO is not required to be online all the time. In order to enable security and privacy, the smart contract is executed in the TEEs. Besides, all interactions, data delegations, and other operations are recorded in the blockchain and can be checked at any time, which realizes the efficient monitoring and auditing of the data. We proved that the security properties, such as confidentiality, anonymity, and verifiable integrity, are ensured during the whole data sharing process. We also simulated the proposed scheme, and the results show that our scheme has a better performance than the related works.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Presidential Foundation of CAEP (Grant No. CX20220001), the Defense Industrial Technology Development Program (JCKY2019602B013), and the Natural Science Foundation (U19A2066).

References

[1] Y. Liu, W. Guo, C.-I. Fan, L. Chang, and C. Cheng, "A practical privacy-preserving data aggregation (3pda) scheme for smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1767–1774, 2019.

[2] H. Shafagh, L. Burkhalter, A. Hithnawi, and D. Simon, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on cloud computing security workshop*, pp. 45–50, Dallas TX USA, November 2017.

[3] B. Matt, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 127–144, Springer, Trondheim, Norway, June 1998.

[4] C. Ran and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 185–194, Alexandria, VA, USA, November 2007.

[5] J. Weng, R. H. Deng, X. Ding, C.-K. Chu, and J. Lai, "Conditional proxy re-encryption secure against chosen-ciphertext attack," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pp. 322–332, Sydney Australia, March 2009.

[6] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343–1354, 2013.

[7] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, vol. 258, pp. 355–370, 2014.

[8] L. Xu, X. Wu, and X. Zhang, "Cl-pre: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM symposium on information, computer and communications security*, pp. 87-88, Seoul, Republic of Korea, May 2012.

[9] B. Huang, Z. Liu, J. Chen, A. Liu, Q. Liu, and Q. He, "Behavior pattern clustering in blockchain networks," *Multimedia Tools and Applications*, vol. 76, no. 19, Article ID 20099, 2017.

[10] M. Zhang, C. Chen, T. Wo, T. Xie, M. Z. A. Bhuiyan, and X. Lin, "Safedrive: online driving anomaly detection from large-scale vehicle data," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2087–2096, 2017.

[11] Y. J. Galteland and S. Wu, *Blockchain-based privacy-preserving fair data trading protocol*, Cryptology ePrint Archive, 2021, https://eprint.iacr.org/2021/1321.

[12] H. Huang, P. Zhu, F. Xiao, X. Sun, and Q. Huang, "A blockchain-based scheme for privacy-preserving and secure sharing of medical data," *Computers & Security*, vol. 99, Article ID 102010, 2020.

[13] 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 Systems Journal*, vol. 16, no. 1, pp. 1685–1696, 2022.

[14] J. He, Z. Dong, R. Guo, Y. Chen, K. Li, and X. Tao, "Efficient identity-based proxy re-encryption scheme in blockchain-assisted decentralized storage system," *International Journal on Network Security*, vol. 23, no. 5, pp. 776–790, 2021.

[15] R. Li, T. Song, Bo Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale internet of things data storage and protection," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 762–771, 2019.

[16] T. McConaghy, R. Marques, A. Müller et al., *Bigchaindb: A Scalable Blockchain Database*, white paper, BigChainDB, Berlin, Germany, 2016.

[17] T. Li, H. Wang, D. He, and J. Yu, "Blockchain-based privacy-preserving and rewarding private data sharing for iot," *IEEE*

Internet of Things Journal, vol. 9, no. 16, Article ID 15138, 2022.

[18] Y. Wang, Z. Su, N. Zhang et al., "Spds: a secure and auditable private data sharing scheme for smart grid based on blockchain," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7688–7699, 2021.

[19] H. Lei, Y. Yan, Z. Bao, Q. Wang, Y. Zhang, and W. Shi, "Sdsbt: a secure multi-party data sharing platform based on blockchain and tee," in *Proceedings of the International Symposium on Cyberspace Safety and Security*, pp. 184–196, Springer, Haikou, China, December 2020.

[20] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.

[21] C. Linnhoff-Popien, R. Schneider, and M. Zaddach, *Digital Marketplaces Unleashed*, Springer, Singapore, 2018.

[22] V. Karande, E. Bauman, Z. Lin, and L. Khan, "Sgx-log: securing system logs with sgx," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 19–30, Abu Dhabi, UAE, April 2017.

[23] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using arm trustzone to build a trusted language runtime for mobile applications," in *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*, pp. 67–80, Salt Lake City, UT, USA, March 2014.

[24] C. Garlati and S. Pinto, "A clean slate approach to linux security risc-v enclaves," in *Proceedings of the Embedded World Conference*, p. 5, Nuremberg, Germany, February 2020.

[25] M. Bowman, A. Miele, M. Steiner, and V. Bruno, "Private Data Objects: An Overview," 2018, https://arxiv.org/abs/1807.05686.

[26] B. V. B. M. Andrea Miele and A. Adesokan, "Private-data-objects," https://github.com/hyperledger-labs/private-data-objects.

[27] R. Cheng, F. Zhang, J. Kos et al., "Ekiden: a platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 185–200, IEEE, Stockholm, Sweden, June 2019.

[28] A. Prashanth Joshi, M. Han, and Y. Wang, "A survey on security and privacy issues of blockchain technology," *Mathematical Foundations of Computing*, vol. 1, no. 2, pp. 121–147, 2018.

[29] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[30] D. Schultz, B. Liskov, and M. Liskov, "Mpss: mobile proactive secret sharing," *ACM Transactions on Information and System Security*, vol. 13, no. 4, pp. 1–32, 2010.

[31] M. Scott, K. McCusker, A. Budroni, and S. Andreoli, "The Miracl Core Cryptographic Library," 2019, https://github.com/miracl/core.