


Research Article

BaDS: Blockchain-Based Architecture for Data Sharing with ABS and CP-ABE in IoT

Yunru Zhang,^{1,2} Debiao He ,^{1,2} and Kim-Kwang Raymond Choo³

¹Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education,
School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

²Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing University of Posts and Telecommunications,
Nanjing 210023, China

³Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249, USA

Correspondence should be addressed to Debiao He; hedebiao@163.com

Received 20 August 2018; Revised 28 September 2018; Accepted 18 October 2018; Published 4 November 2018

Guest Editor: Georgios Kambourakis

Copyright © 2018 Yunru Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Internet of Things (IoT) and cloud computing are increasingly integrated, in the sense that data collected from IoT devices (generally with limited computational and storage resources) are being sent to the cloud for processing, etc., in order to inform decision making and facilitate other operational and business activities. However, the cloud may not be a fully trusted entity, like leaking user data or compromising user privacy. Thus, we propose a privacy-preserving and user-controlled data sharing architecture with fine-grained access control, based on the blockchain model and attribute-based cryptosystem. Also, the consensus algorithm in our system is the Byzantine fault tolerance mechanism, rather than Proof of Work.

1. Introduction

The Internet of Things (IoT) has many applications in a wide range of industries and settings, such as smart homes and intelligent transportation systems, as well as in consumer applications (e.g., medical and health-care equipment) [1, 2]. One typical role of IoT devices (e.g., sensors and smart devices) is to collect and transmit (the collected) data via the Internet, like further processing and statistical analysis. However, IoT devices are generally resource-constrained, for example, having limited computational and storage resources. Thus, there has been a trend for integrating IoT and the cloud, to which data storage, processing, and sharing functionalities are being outsourced [3, 4].

As shown in Figure 1, *Owner1* can store and share the collected data with *Owner2* via the cloud to minimize costs. However, there is a risk that data and user's privacy may be leaked and compromised since the cloud is not fully trusted (i.e., semitrusted). Although there exists many privacy-preserving data processing solutions (e.g., utilizing cryptographic tools based on access control policies [5, 6]) for cloud storage systems, these approaches are vulnerable to

attacks at the cloud end (e.g., access control policies may be tampered or deleted by a malicious cloud service provider or its employee).

Blockchain is a distributed ledger technology that underpins Bitcoin [7] and has been used in many other decentralized applications, such as digital currency [8, 9], data storage [10, 11], data provenance [12], Internet of Things [13–16], and so on. In this paper, we posit the potential of integrating blockchain with attribute-based cryptosystems [17, 18] in the design of a privacy-preserving and user-controlled solution to IoT data sharing. In other words, users can independently decide who can share their data without compromising data and identity privacy. Specifically, in our proposed BaDS (Blockchain-Based Architecture for Data Sharing with ABS and CP-ABE in IoT) architecture:

- (i) IoT data are first encrypted (e.g., AES). Then, we integrate smart contract technology with an attribute encryption scheme [19] to realize its fine-grained sharing. The access policies are set on the encrypted key (the encrypted key are encrypted by attributes,

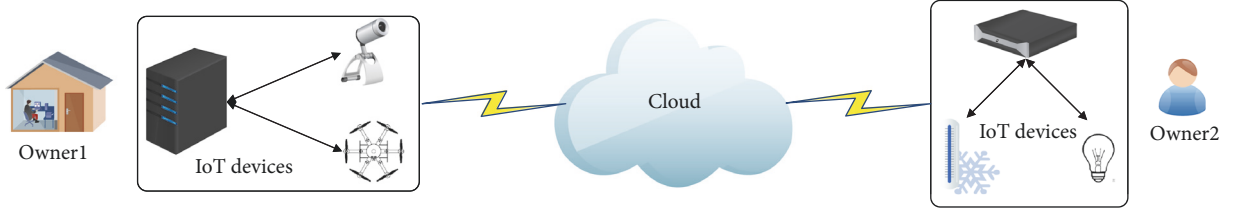


FIGURE 1: IoT data storage and sharing in a cloud-based model.

ABE) to decide who can obtain this encrypted key to decrypt the ciphertext.

- (ii) The smart contract used in our architecture is to ensure the scalability of access control table. All the data sharing (or access) requests in the system interact with smart contracts through transactions (e.g., smart contract in Section 3.5).

The rest of the paper is organized as follows: In Section 2, we present the relevant cryptographic techniques, monotone span program, the network model, and security requirements. In Section 3, we briefly introduce ABS, CP-ABE, PACT, PBFT, and smart contract. After that, we describe our BaDS architecture and its security analysis in Section 4. In Section 5, we describe our evaluation of the proposed architecture, prior to concluding the paper in the last section.

2. Preliminaries

In this section, we will introduce the cryptographic techniques, monotone span program, the network model of our propose BaDS architecture, and the security requirements that need to be satisfied.

2.1. Bilinear Pairings. We define G_1 and G_2 as two additive cyclic groups on elliptic curve $F(p)$, G_T as a multiplication cyclic group. Let q be a big prime number, which is the order of G_1 , G_2 , and G_T . $e : G_1 \times G_2 \rightarrow G_T$ denote a bilinear map. Suppose that the generators of G_1 and G_2 are P and Q ; g is the element that P and Q map to G_T . Thus, the map e is a bilinear pairing on condition that e satisfies the following properties:

- (i) **Bilinearity.** Given any two elements $a, b \in Z_q^*$, and $\forall X \in G_1, \forall Y \in G_2$, there is $e(a \cdot X, b \cdot Y) = e(X, Y)^{a \cdot b}$.
- (ii) **Nondegenerate.** There exists at least one element X which satisfies $e(X, X) \neq 1$.
- (iii) **Efficient Computability.** Given any two elements $\forall X \in G_1, \forall Y \in G_2$, there exists at least one efficient algorithm to compute $e(X, Y)$.

We define the computationally hard mathematical problems.

- (i) **Discrete Logarithm (DL) Problem.** Given an element $X \in G_1$ or $x \in G_T$, for any P.P.T (probability polynomial time) attacker, it is computationally hard to calculate $\tau \in Z_q^*$ which satisfies $X = \tau \cdot P$ or $x = g^\tau$.

- (ii) **Computational Diffie-Hellman (CDH) Problem.**

Given a tuple (g, e, P, Q, aP, bQ) in which $(a, b) \in Z_q^*$, P, Q and g are the generators of G_1, G_2 , and G_T , respectively. The purpose of **CDH** problem is to compute $\omega = g^{ab} \in G_T$, in which $(a, b) \in Z_q^*$ are unknown.

2.2. Monotone Span Program. Let $\Gamma : \{0, 1\}^n \rightarrow \{0, 1\}$ as a monotone Boolean function. For an $l \times t$ matrix M over a field F and every $(x_1, \dots, x_n) \in \{0, 1\}^n$, a monotone span program is defined as follows:

$$\begin{aligned} \Gamma(x_1, x_2, \dots, x_n) = 1 &\iff \\ \exists \vec{v} \in F^{l \times t} : M \cdot \vec{v} &= [1, 0, 0, \dots, 0] \end{aligned} \quad (1)$$

in which $\forall i : x_{a(i)} = 0 \implies v_i = 0$ and the labeling function $a : [l] \rightarrow [n]$. That means $\Gamma(x_1, x_2, \dots, x_n) = 1$ if and only if the index to the rows of matrix spans the vector $[1, 0, 0, \dots, 0]$. We say the length and width of span program are l and t , and the size of it is $l + t$.

2.3. Network Model. Our BaDS architecture consists of the following participants: **IoT devices**, **Data Owner**, **Blockchain Network**, and **Cloud**; see Figure 2.

- (i) **IoT Devices.** IoT devices collect data and send data to the network layer (e.g., cloud or some other applications). Such devices are also responsible for data acquisition, preliminary processing, encryption (if they can support the encryption), and transmission. The devices can usually remotely request access and handle the commands. When the devices need to request data from other devices, they should publish a corresponding request to the cloud or the data owner.
- (ii) **Data Owner.** There is a very large number of data owners, who are divided into administrators and ordinary data owners. The administrators are responsible for vetting the participants. When data owner receives an access data request from other IoT device, he/she should authenticate the identity before responding to the request accordingly.

- (iii) **Blockchain Network.** In this architecture, we adopt a permissioned model (e.g., hyperledger fabric). Specifically, its security is guaranteed under the assumption that most participants are honest and the difficult problems. In other words, the average time an attacker

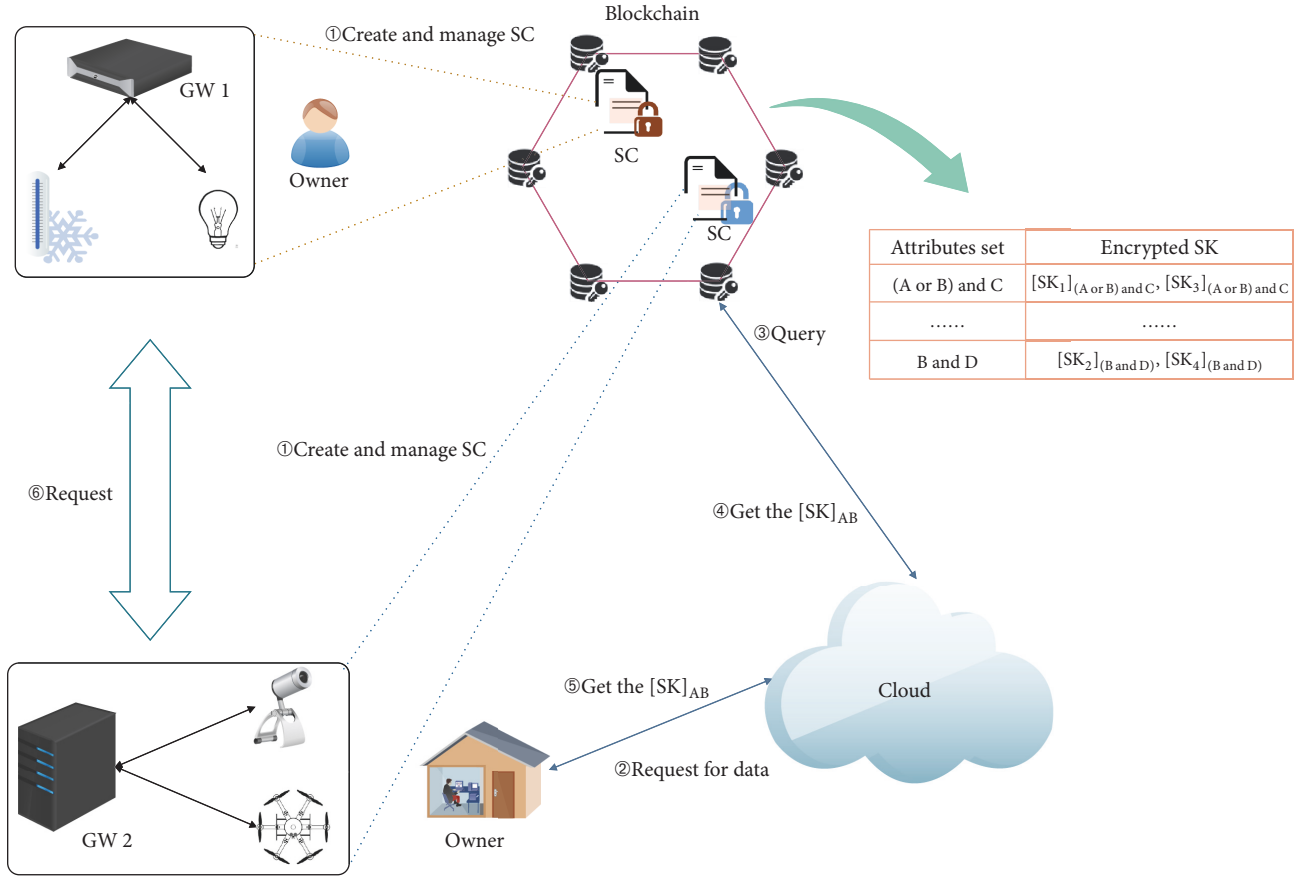


FIGURE 2: IoT device data sharing system.

can solve the problem is much less than the time it takes to disseminate information over the network. The consensus algorithm in our system is the Byzantine fault tolerance mechanism **PBFT**, rather than Proof of Work (**POW**) used in Bitcoin. The fabric contains verification nodes (to verify the transaction) and ordering nodes (to pack the verified transaction into the block). When the nodes receive the transaction (request from cloud or data owner), they will verify and pack them into the blockchain.

- (iv) **Cloud.** It is used to store the encrypted devices' data, and sends a corresponding request transaction to the blockchain network to query the permission of the device, when the cloud receives a request from IoT device. That means the cloud monitors the blockchain network and responds to the requested data.

In the BaDS architecture, the following steps are undertaken to request data between devices.

At first, owner *A* sends the encrypted devices' data to the cloud and generates the responding access control table in the smart contract. When other device (e.g., device belonging to owner *B*) wants to access the data, it invokes the getPACT algorithm in the smart contract to obtain the predicates. If its attributes satisfy the predicates, then it sends a responding transaction to the smart contract with an attribute-based

signature. If the signature can be successfully verified, then the device can receive the encrypted private key (CP-ABE).

Then, the device sends a request containing an attribute-based signature to the cloud to obtain the requested data. On receiving the request, the mutual authentication and session key between the cloud and device will be established by executing the existing authentication key agreement protocol [20]. After both sides have mutual authenticated each other, the device can get the encrypted data through the "secure channel" protected by the session key.

If the signature cannot be verified, it implies that the attributes of the device do not satisfy the policy of the data. Thus, owner *B* should send a request for data access to owner *A*. Both owners will authenticate each other and generate a session key through the existing authentication key agreement protocol, and the session key is used to guarantee the subsequent session. If owner *A* permits the access from the device of *B*, he/she will send a transaction to the smart contract to update the access control table and the device can obtain the data from the cloud as before. Otherwise, *A* rejects the request.

2.4. Security Requirement. Based on recent literatures [20–22], the blockchain-based architecture for data sharing with ABS and CP-ABE needs to satisfy the following security requirements:

- (1) **Confidentiality.** To protect the privacy of data, in this architecture, only devices which satisfy the attribute policy can access the data and get the corresponding decryption key.
- (2) **Fine-Grained Access Control.** The data manager or authority generate the corresponding access policy for their data, and they can grant or revoke devices' access on a fine-grained basis, by modifying the access attributes.
- (3) **Mutual Authentication.** To protect the safety of participants, our system should provide mutual authentication. The participants should authenticate their communicating partner.
- (4) **User Anonymity.** To preserve privacy, the architecture should protect the device's anonymity. Even if the adversary analyzes a series of transactions, (s)he cannot learn the devices' real identity.
- (5) **Impersonation Attack Resilience.** If the adversary impersonates a legitimate device and sends a request to the cloud, it cannot be authenticated due to the invalid attribute signature.
- (6) **Collision Attack Resilience.** There is an extremely small possibility of generating two identical blocks at the same time. Thus, the system should resist collision attack.
- (7) **Man-in-the-Middle Attack Resilience.** The device can identify and abandon the messages transmitted in the open environment, which have been intercepted or replaced by the adversary.
- (8) **Link Attack Resilience.** Even if the adversary links multiple transactions which use the same address or public key, the adversary also cannot find users' private messages.

3. Definitions and Security Model

We briefly introduce the attribute-based signature, ciphertext-policy attribute-based encryption, permission access control table to generate the attribute policy, and PBFT (the consensus algorithm) used in our architecture, in this section.

3.1. Attribute-Based Signature (ABS). In the attribute-based signature scheme, the devices are tagged with a set of attributes whose certificates are issued by an attribute center [23]. Due to the fact that ABS scheme can provide fine-grained access control, we use ABS in our architecture to replace the original ECDSA signature in the blockchain.

A signature consists of $s+2$ elements, where s is the width of the monotone span program of the claim-predicate [24]. The maximum width of monotonic span program is defined as q_m , and $\mathbb{A} = \mathbb{Z}_p^*$, in which p is a big prime number and also the order of the cyclic group.

- (i) **ABS.PSetup.** Let G_1 , G_2 , and $e : G_1 \times G_2 \rightarrow G_T$ denote two cyclic groups and a bilinear map, respectively, where the order of cyclic groups is p .

Suppose g and h_i are the generators of G_1 and G_2 , where $i \in [0, q_m]$. Choose a hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$. Thus, the public parameters are $\text{Param} = \{G_1, G_2, H, g, h_0, \dots, h_{q_m}\}$.

- (ii) **ABS.MSetup.** Randomly choose four numbers $l_0, l, m, n \in \mathbb{Z}_p^*$ and compute: $N = g^n$, $L_0 = h_0^{l_0}$, $L_j = h_j^l$, and $M_j = h_j^m$, $j \in [1, q_m]$. Thus, the master key is $\text{MSK} = \{l_0, l, m\}$, and the public key is $\text{MPK} = \{N, L_0, \dots, L_{q_m}, M_1, \dots, M_{q_m}\}$.
- (iii) **ABS.Gen.** Input the master key MSK and an attribute set $\mathcal{A} \subseteq \mathbb{A}$. Choose a random generator K_{base} in cyclic group G_1 and compute $K_0 = K_{\text{base}}^{l_0^{-1}}$ and $K_u = K_{\text{base}}^{(l+mu)^{-1}}$, $u \in \mathcal{A}$. Thus, the device private key is $\text{SK}_{\mathcal{A}} = \{K_{\text{base}}, K_0, K_u\}$, $u \in \mathcal{A}$.
- (iv) **ABS.Sign.** Input the public key MPK , device private key $\text{SK}_{\mathcal{A}}$, the message m , and a monotone Boolean function γ , in which $(\gamma(\mathcal{A}) = 1)$. $\gamma \rightarrow \text{MN} \in (\mathbb{Z}_p)^{x \times y}$, $u : [x] \rightarrow \mathbb{A}$, and the vector \vec{v} meets the assignment \mathcal{A} . Compute $\mu = H(m \parallel \gamma)$ and choose the random numbers $\beta_0 \in \mathbb{Z}_p^*$ and $\beta_1, \dots, \beta_x \in \mathbb{Z}_p$. After that compute $C = K_{\text{base}}^{\beta_0}$, $F = K_0^{\beta_0}$, $S_i = (K_{u(i)}^{\beta_i})^{\beta_i} \cdot (Ng^{\mu})^{\beta_i}$, and $R_i = \prod_{j=1}^x (L_j \cdot M_j^{u(i)})^{T_{ij} \cdot \beta_i}$, $\forall i \in [x]$ and $\forall j \in [y]$. The signature is $\sigma = \{C, F, S_i, R_j\}$, $\forall i \in [x]$ and $\forall j \in [y]$.
- (v) **ABS.Veri.** Input the public key MPK , the signature σ , the message m , and the monotone Boolean function γ . First compute $\gamma \rightarrow \text{MN} \in (\mathbb{Z}_p)^{x \times y}$, $u : [x] \rightarrow \mathbb{A}$, and $\mu = H(m \parallel \gamma)$. If $C = 1$, then output 0. Otherwise, checks $e(F, L_0) = e(C, h_0)$ and

$$\prod_{i=1}^x e \left(S_i, (L_j M_j^{u(i)})^{T_{ij}} \right) = \begin{cases} e(C, h_1) e(Ng^{\mu}, R_1), & j = 1 \\ e(Ng^{\mu}, R_j), & j > 1, \end{cases} \quad (2)$$

If they are all equal, then return 1; otherwise, return 0.

3.2. Ciphertext-Policy Attribute-Based Encryption (CP-ABE). Attribute-based encryption has been used to share data with some target devices which have specified attributes [19]. The data owner can make policies (s)he wishes to share the data with. The users will be assigned a secret key associated with the attributes, and they can decrypt (or access) the shared data if their attributes "satisfy" the predicates [25].

- (i) **ABE.Setup.** Input an attribute set \mathbb{A} . Choose two multiplicative cyclic groups G_3 and G_T , whose order is p . Let $e : G_3 \times G_3 \rightarrow G_T$ denote a bilinear map. Suppose k is a generator of G_3 , and $f_1, \dots, f_{|\mathbb{A}|}$ are $|\mathbb{A}|$ random elements in group G_3 . Select two random numbers $s, a \in \mathbb{Z}_p$, and compute the public parameters $\text{Param} = \{k, G_3, f_1, \dots, f_{|\mathbb{A}|}\}$ and the master private key $\text{ESK} = k^s$.

- (ii) **ABE.KeyGen.** Input the master private key ESK and an attribute set \mathcal{A} . Choose a random number $t \in \mathbb{Z}_p$, and compute $sk = k^s k^{at}$, $T = k^t$, $sk_i = f_i^t (\forall i \in \mathcal{A})$. The private key is $SK = (sk, T, sk_i)$; the public key is $PK = \{e(k, k)^s, k^a\}$.
- (iii) **ABE.Encrypt.** Input the public parameters $Param$, the public key PK , the message m , and an Linear Secret Sharing Scheme (LSSS) access structure (M, β) . M is a $x \times y$ matrix and β is a function that links rows of M with attributes. Choose a vector $\vec{v} = (l, b_2, \dots, b_y)^T \in \mathbb{Z}_p^y$, and compute $\gamma_i = \vec{v} \cdot M_i$, M_i is the i th row of matrix M . Select x random numbers $n_1, \dots, n_x \in \mathbb{Z}_p$, and compute $C = Me(k, k)^{sl}$, $C' = k^l$, $(C_1 = k^{a\gamma_1} f_{\beta(1)}^{-n_1}), \dots, (C_x = k^{a\gamma_x} f_{\beta(x)}^{-n_x}), D_x = k^{n_x}$. The ciphertext is $CT = (C, C', (C_1, D_1), \dots, (C_x, D_x))$.
- (iv) **ABE.Decrypt.** Input the private key SK for attribute set \mathcal{A} and the ciphertext CT for (M, β) . Suppose \mathcal{A} satisfies (M, β) and define $U = \{i : \beta(i) \in \mathcal{A}\}$. Let $\rho_i \in \mathbb{Z}_p$, $i \in U$. and $\sum_{i \in U} \rho_i \gamma_i = l$. Decrypt the ciphertext:

$$\frac{e(C', sk)}{\left(\prod_{i \in U} (e(C_i, T) e(D_i, sk_{\beta(i)}))^{\rho_i}\right)} = \frac{e(k, k)^{sl} e(k, k)^{alt}}{\prod_{i \in U} e(k, k)^{t a \gamma_i \rho_i}} \quad (3)$$

$$= e(k, k)^{sl}$$

3.3. Permission Access Control Table PACT. We generate a permission access control table (PACT) to achieve fine-grained access control by using smart contract. The owner first deploys smart contract with the access control table in the blockchain. This allows other devices to request and/or access data when their attributes satisfy the predicates. For example, “(A or B) and C” mapping to “[SK_1]_{(AorB)andC}” and “[SK_3]_{(AorB)andC}” mean the devices which satisfy the attribute “(A or B) and C” can access the encrypted private keys of device with identities 1 and 3. Only the device (or smart contract) owner can update the PACT by calling the smart contract function.

3.4. Practical Byzantine Fault Tolerance (PBFT). The consensus algorithm used in this paper is Practical Byzantine Fault Tolerance (PBFT). We assume that there are a total of $3f + 1$ nodes in the system, where f is the maximum number of nodes that may be failed. When more than $2f + 1$ normal authorized nodes confirm the transaction, the authorized nodes come to a consensus. This means that users will eventually receive replies from authorized nodes pertaining to their requests.

This algorithm is suitable for asynchronous systems such as the Internet. It contains important optimization functions that enable it to be executed efficiently. Here, we introduce the working process of PBFT, which consists of the following five phases: **Request**, **Preprepare**, **Prepare**, **Commit**, and **Replay**.

- (i) **Request.** When the leader is found not to be honest, the other replica is elected as a new leader by the algorithm. The primary sends a request to a replica; here it is replica 0.

- (ii) **Preprepare.** When replica 0 receives the request, it broadcasts a preprepared message to other replicas.
- (iii) **Prepare.** When the other replicas receive the preprepared message, if they accept, they broadcast the prepare message to all the other replicas and add preprepare and prepare messages into their logs. Otherwise, they do nothing.
- (iv) **Commit.** When replicas receive more than a certain number ($2f$) of prepare messages during the **Prepare** phase, it enters the **Commit** phase. The replicas broadcast commit message.
- (v) **Replay.** If more than $2f + 1$ replicas accept the commit message, it means that there is a replica receiving more than $2f + 1$ commit messages. After completing the request operation, each replica sends a replay message to the primary node.

Both **Preprepare** and **Prepare** phases are used to ensure the ordering of the request. The consensus algorithm does not rely on the orderly propagation of messages, so replicas can submit requests in a disorderly manner. Because each replica backs up the message log in the preprepare, prepare, and commit phases, the corresponding requests can be executed in order.

3.5. Smart Contract. The concept of smart contract is introduced in 1994 by Nick Szabo and defined as “a computerized transaction protocol that executes the terms of a contract” [26]. Smart contracts are autonomous scripts stored on blockchain and have unique addresses. The creator can compile, deploy, and update his/her smart contract, and the output is recorded in blockchain network as a transaction. In our architecture, we use smart contract to manage PACT. The devices can send a request to the cloud with its signature, and the cloud interacts with the smart contract to verify the signature and retrieve the policy.

4. Proposed BaDS Architecture

In the proposed BaDS architecture, the devices use their attribute private key to sign the data request. The architecture comprises **Initialization**, **Request and Transaction for Cloud**, and **Request and Transaction for Owner**. Let us assume that a device belonging to owner A wants to access the data of owner B. The steps below are carried out among the parties.

- (1) **Initialization.** In this phase, the authority or system manager selects the system private key and computes the parameters by executing the following steps.
 - (i) **ABS.Initialization.** As explained in Section 3.1, select a maximum width of monotonic span program q_m and cyclic groups G_1, G_2 . Generate the public parameters $Param = \{G_1, G_2, H, g, h_0, h_{q_m}\}$. Choose the master key $MSK = \{l_0, l, m\}$, and compute the public key $MPK = \{N, L_0, \dots, L_{q_m}, M_1, \dots, M_{q_m}\}$ by

calling ABS.PSetup and ABS.MSetup . When each device registers on the system for the first time, based on its attribute tags \mathcal{A} , manager generates the private key $SK_{\mathcal{A}} = \{K_{base}, K_0, K_u\}$ for device using the ABS.Gen algorithm.

- (ii) **ABE.Initialization.** Select a cyclic group G_3 and a generator k . Choose the master key $ESK = k^s$, and generate the public parameters $Param = \{k, G_3, f_1, \dots, f_{|\mathcal{A}|}\}$ by calling ABE.Setup . As explained in Section 2.3, the assumption is that there are π IoT devices, two device managers (e.g., owner A and owner B) and a cloud in the system.

When the nodes register on the system for the first time, based on their attribute tags \mathcal{A} , manager generates their public and private key pairs (PK_j, SK_j) ($j = 1, \dots, \pi, oa, ob, c$) using ABE.KeyGen . $PK_j = \{e(k, k)_j^s, k_j^a\}$, and $SK_j = \{sk_j, T_j, sk_{ij}\}$, where $sk_j = k_j^s k_j^{at}$, $T_j = k_j^t$, $sk_{ij} = f_{ij}^t (\forall i \in \mathcal{A})$.

- (iii) **Contract Deployment.** First, we generate a smart contract which is designed to achieve permission access control table. Then we compile and deploy it on the blockchain, after that the smart contract will have its own address (e.g., PID). The access control table is made up of the device access policy and corresponding predicates. The data sharing private keys are encrypted with relevant attributes.
- (2) **Request and Transaction for Cloud.** In this phase, a device belonging to owner A invokes the algorithm in the smart contract for sharing data belonging to other devices. It executes as follows.

- (i) **Query.** The device invokes the getPACT algorithm in smart contract to get the corresponding predicate of the target device's access policy. And it checks whether its attributes can satisfy the predicate or not. If yes, then it sends a transaction to smart contract. Otherwise, the device requests permission from owner.
- (ii) **Transaction.** The device prepares and constructs the corresponding transaction based on its request. For instance, a transaction consists of $\{to, from, value\}$ and other parts, "to" is filled with the address of smart contract that the device wants to call, and "from" is filled with the device's address. After all the fields are constructed, the device uses its attribute private key $SK_{\mathcal{A}}$ to sign the transaction and broadcasts it to the blockchain network.
- (iii) **State.** The other nodes invoke the ABS.Ver algorithm to verify the attribute-based signature in it, when they receive the broadcast transaction. The architecture uses the PBFT consensus mechanism to achieve consensus, and the

transaction can be recorded in the blockchain network, only if there are at least two-thirds nodes that have accepted it.

- (iv) **Response.** The smart contract returns the attribute-based encrypted private key in the access policy as a response message to the device, after the transaction is recorded in the blockchain network. The device can use its attribute private key SK_j to get the private key. Then, the device sends a request to the cloud. The two parties (between device and cloud) should authenticate each other using the existing authentication protocol [20] before the cloud deals with the request. Finally, the device obtains the data from the cloud and uses the private key to decrypt the data.
- (3) **Request and Transaction for Owner.** In this phase, the device requests permission from data owner. It executes as follows.
- (i) **Request.** The device invokes the getPACT algorithm in smart contract to get the corresponding predicate of the target device's access policy. If its attributes cannot satisfy the predicate, then it requests for permission from the owner. Both device and owner should authenticate each other using the existing authentication protocol [20], prior to the owner dealing with the request. If the owner allows the device to access the data, then he/she calls the smart contract to modify the access control table by transaction. Otherwise, the device cannot access the data.
 - (ii) **Permission Update.** As described in Section 3.3, we use permission access control table in smart contract to achieve fine-grained access control. The data owner can invoke the UpdatePACT algorithm in smart contract to add, delete, and modify the access policies and the predicates. Only if the device's attributes satisfy the predicates in PACT can it obtain the encrypted key from access policies.
 - (iii) **Access.** After the data owner has modified PACT, the device sends a request to cloud and performed as described in **Request and Transaction for Cloud**.

4.1. Security Analysis. In this section, we analyze how the architecture is resilient to the following typical security and privacy attacks.

- (1) **Confidentiality.** The permission access control table (PACT) is generated to restrict access; if an adversary sends request to cloud or smart contract, the request will be rejected owing to his/her invalid signature. Thus, only authorized (satisfied attribute policy) devices can access the data and get the corresponding decryption key.

- (2) **Fine-Grained Access Control.** The attribute-based signature provides fine-grained access control. In other words, the data manager or authority generate policies (some attributes set), and only devices satisfying the policies can access the data. In addition, the manager can grant or revoke device access by modifying the policy.
- (3) **Mutual Authentication.** Before the device can communicate securely with the cloud or data owner, they will confirm the identity of each other by their signatures. Any probabilistic polynomial time adversary cannot forge a valid signature due to the underpinning DL problem. Hence, mutual authentication can be achieved between the device and the cloud or data owner.
- (4) **User Anonymity.** In the BaDS architecture, we use attribute-based signature and encryption to protect the devices' real identities. All the message transmitted in the open channel are signed or encrypted by some attributes or session key. Thus, when verifying the signature or decrypting the information, only the attributes public or secret keys are needed. Hence, the architecture can protect user anonymity.
- (5) **Impersonation Attack.** As discussed earlier, mutual authentication between devices and cloud or data owner is achieved in the BaDS architecture. If the adversary impersonates a legitimate device and sends a request to cloud, it cannot be authenticated, and the manager can revoke access of the malicious or compromised device. That is, only a legitimate device can generate a valid signature.
- (6) **Collision Attack Resistance.** In the BaDS architecture, we use the PBFT consensus algorithm to record new blocks, which effectively avoids collisions of blocks.
- (7) **Man-in-the-Middle Attack Resistance.** Man-in-the-middle attack means the adversary can intercept and replace the encrypted data transmitted in the open environment. Suppose that if an adversary modifies other response message, the device can identify them due to the use of the attribute signature and session key encryption; thus we can say that the adversary cannot modify the transaction message.
- (8) **Link Attack Resistance.** Link attack is defined as the adversary can link multiple transaction, which use the same address or public key, to find users' private messages. Similar to our explanation for man-in-the-middle attack, all messages transmitted in the open environment are signed or encrypted by the attributes and session key. Even if the adversary obtains session key, he/she cannot extract something useful due to the attribute encryption of real device's data.

5. Performance Analysis

In this section, we implement the BaDS architecture and analyze the computation cost of smart contract based on

TABLE 1: Simulation platform.

Operating System	Ubuntu 16.04
CPU	Intel (R) Core (TM) i7-6700 CPU @ 3.40 GHZ
Memory	3 GB RAM
Configuration	go-ethereum nodejs npm truffle

Ethereum (<https://www.ethereum.org/>). The blockchain platform allows one to write smart contract with a special language, and compile and deploy it to the blockchain network. The smart contracts are autonomous scripts stored on blockchain and have unique addresses. Thus, it can be regarded as a database in which function can be called by sending a transaction with corresponding parameters.

We define Auth.KA as authentication key agreement algorithm and PACT.deploy and SC.deploy as the deployment of permission access control table in smart contract and the deployment of smart contract in blockchain, respectively. getPACT and UpdataPACT denote invoking the corresponding get and update functions in smart contract. The operations needed at each phase in this architecture is shown in Table 2. We then evaluate the operation computation cost of those algorithm (e.g., ABS, CP-ABE) by using the pairing-based library and GNU multiple precision arithmetic library. Table 3 shows the computation cost of attribute-based signature and encryption algorithms.

We publish the smart contract on a private Ethereum network, which we constructed by ourselves, then we can compute the time of deploying and invoking a smart contract. Publishing transactions on private chain does not need transaction fees and has the same accurate results as public chains. Table 1 presents the information of the simulation platform. *Web3j* is used to evaluate the time cost of publishing a designed smart contract. However, the existing Ethereum platform does not provide ABS and ABE algorithm; in this research, we only use the smart contract to realize management of policies and execute the signing and encrypting in the external environment.

6. Conclusion

We proposed a novel blockchain-based architecture for data sharing with attribute-based cryptosystem (BaDS) in this paper. The architecture can achieve privacy-preserving, user-self-controlled data sharing, and decentralization by using blockchain and several attribute-based cryptosystems. Specifically, ABS and CP-ABE provide the capability for fine-grained access control. We introduced the security requirements of the proposed BaDS architecture and then explained how the proposed BaDS architecture satisfies the security requirement. We also implement the BaDS architecture and analyze its computation cost.

TABLE 2: Operations needed at each phase in BaDS.

Phase	Cryptographic Algorithms	Smart Contract Publication	Smart Contract Trigger
Initialization	ABS.PSetup + ABS.MSetup +ABS.Gen + ABE.Setup + ABE.KeyGen + ABE.Encrypt	PACT.deploy + SC.deploy	Null
Request and Transaction for cloud	Auth.KA + ABE.Decrypt + ABS.Sign + ABS.Veri	Null	getPACT
Request and Transaction for Owner	Auth.KA + 2*ABS.Sign + 2*ABS.Veri	Null	getPACT + UpdataPACT

TABLE 3: Computation cost(s) of cryptographic algorithms.

ABS				
Algorithm	ABS.PSetup & ABS.MSetup	ABS.Gen	ABS.Sign	ABS.Veri
Max Time	0.17939	0.068969	0.136605	0.18168
Min Time	0.049207	0.021929	0.055461	0.050144
Average Time	0.058182	0.028793	0.069066	0.064048
CL-MRE				
Algorithm	ABE.Setup	ABE.KeyGen	ABE.Encrypt	ABE.Decrypt
Max Time	0.161446	0.058168	0.164257	0.234115
Min Time	0.037582	0.017119	0.050459	0.025966
Average Time	0.045421	0.02222	0.05951	0.035304

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

The work was supported in part by the National Key Research and Development Program of China (no. 2018YFC1315400), the National Natural Science Foundation of China (nos. 61572370, 61572379, and 61501333), and the fund of the Jiangsu Key Laboratory of Big Data Security & Intelligent Processing (no. BDSIP1807).

References

- [1] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A Survey on Security and Privacy Issues in Internet-of-Things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.
- [2] H. Debiao, K. Neeraj, W. Huaqun, L. Wang, K. K. R. Choo, and A. Vinel, "A provably-secure cross-domain handshake scheme with symptoms-matching for mobile healthcare social network," *IEEE Transactions on Dependable & Secure Computing*, vol. 15, no. 4, pp. 633–645, 2018.
- [3] Y. Yu, J. Ni, M. H. Au, Y. Mu, B. Wang, and H. Li, "Comments on a public auditing mechanism for shared cloud data service," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 998–999, 2015.
- [4] H. Debiao, K. Neeraj, K. M. Khurram, L. Wang, and J. Shen, "Efficient privacy-aware authentication scheme for mobile cloud computing services," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1621–1631, 2018.
- [5] W. Ding, Z. Yan, and R. Deng, "Privacy-preserving data processing with flexible access control," *IEEE Transactions on Dependable and Secure Computing*, 2017, In press.
- [6] K. Yang, X. Jia, and K. Ren, "Attribute-based fine-grained access control with efficient revocation in cloud storage systems," in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIACCS '13)*, pp. 523–528, Hangzhou, China, May 2013.
- [7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Consulted, 2008.
- [8] I. Miers, C. Garman, M. Green, and A. D. Rubin, "ZeroCoin: Anonymous distributed e-cash from bitcoin," in *Proceedings of the 34th IEEE Symposium on Security and Privacy, SP 2013*, pp. 397–411, Berkeley, Calif, USA, May 2013.
- [9] H. Yining, A. Manzoor, P. Ekparinya et al., "A delay-tolerant payment scheme based on the ethereum blockchain," 2018.
- [10] H. Kopp, D. Mödinger, F. Hauck, F. Kargl, and C. Bösch, "Design of a privacy-preserving decentralized file storage with financial incentives," in *Proceedings of the 2nd IEEE European Symposium on Security and Privacy Workshops, EuroS and PW 2017*, pp. 14–22, Paris, France, April 2017.
- [11] H. Kopp, C. Bösch, and F. Kargl, "KopperCoin – a distributed file storage with financial incentives," in *Information Security Practice and Experience*, vol. 10060 of *Lecture Notes in Computer Science*, pp. 79–93, Springer International Publishing, Cham, 2016.
- [12] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A Blockchain-Based Data Provenance Architecture in Cloud Environment with Enhanced Privacy and Availability," in *Proceedings of the 17th IEEE/ACM International*

Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017, pp. 468–477, Madrid, Spain, May 2017.

- [13] M. Banerjee, J. Lee, and K. R. Choo, “A blockchain future for internet of things security: a position paper,” *Digital Communications and Networks*, vol. 4, no. 3, pp. 149–160, 2018.
- [14] C. Lin, D. He, X. Huang, K. R. Choo, and A. V. Vasilakos, “BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0,” *Journal of Network and Computer Applications*, vol. 116, pp. 42–52, 2018.
- [15] S. Huh, S. Cho, and S. Kim, “Managing IoT devices using blockchain platform,” in *Proceedings of the 19th International Conference on Advanced Communications Technology, ICACT 2017*, pp. 464–467, Bongpyeong, Republic of Korea, February 2017.
- [16] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, “Smart Contract-Based Access Control for the Internet of Things,” *IEEE Internet of Things Journal*, 2018, In press.
- [17] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 89–98, Alexandria, Va, USA, November 2006.
- [18] K. H. Maji, M. Prabhakaran, and M. Rosulek, “Attribute-based signatures,” *IACR Cryptology ePrint Archive*, vol. 2010, article 595, 2010.
- [19] B. Waters, “Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization,” in *PKC 2011: Public Key Cryptography – PKC 2011*, Lecture Notes in Computer Science, pp. 53–70, Springer, Berlin, Germany, 2011.
- [20] D. He, S. Zeadally, N. Kumar, and W. Wu, “Efficient and Anonymous Mobile User Authentication Protocol Using Self-Certified Public Key Cryptography for Multi-Server Architectures,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2052–2064, 2016.
- [21] N. Z. Aitzhan and D. Svetinovic, “Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2018.
- [22] Q. Feng, D. He, S. Zeadally, N. Kumar, and K. Liang, “Ideal lattice-based anonymous authentication protocol for mobile devices,” *IEEE Systems Journal*, pp. 1–11.
- [23] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, “Attribute-based signature and its applications,” in *Proceedings of the 5th ACM Symposium on Information, Computer and Communication Security (ASIACCS '10)*, pp. 60–69, Beijing, China, April 2010.
- [24] H. K. Maji, M. Prabhakaran, and M. Rosulek, “Attribute-based signatures,” in *CT-RSA 2011: Topics in Cryptology – CT-RSA 2011*, vol. 6558 of *Lecture Notes in Computer Science*, pp. 376–392, Springer, Heidelberg, Germany, 2011.
- [25] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in *Advances in Cryptology – EUROCRYPT 2005*, vol. 3494 of *Lecture Notes in Computer Science*, pp. 457–473, Springer, Berlin, Germany, 2005.
- [26] N. Szabo, “The idea of smart contracts,” *Nick Szabos Papers and Concise Tutorials*, vol. 6, 1997.

