sensors

MDPI

Article

Blockchain-Based Data Access Control and Key Agreement System in IoT Environment

JoonYoung Lee [1], MyeongHyun Kim [1], KiSung Park [2], SungKee Noh [2], Abhishek Bisht [3], Ashok Kumar Das [3] and Youngho Park [1,4,*]

[1] School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Republic of Korea; harry250@knu.ac.kr (J.L.); kimmyeong123@knu.ac.kr (M.K.)
[2] Decentralized Network Research Section, Electronics and Telecommunications Research Institute, Daejeon 34129, Republic of Korea; ks.park@etri.re.kr (K.P.); sknoh@etri.re.kr (S.N.)
[3] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India; abhishek.bisht@research.iiit.ac.in (A.B.); iitkgp.akdas@gmail.com or ashok.das@iiit.ac.in (A.K.D.)
[4] School of Electronics Engineering, Kyungpook National University, Daegu 41566, Republic of Korea
* Correspondence: parkyh@knu.ac.kr; Tel.: +82-53-950-7842

Abstract: Recently, with the increasing application of the Internet of Things (IoT), various IoT environments such as smart factories, smart homes, and smart grids are being generated. In the IoT environment, a lot of data are generated in real time, and the generated IoT data can be used as source data for various services such as artificial intelligence, remote medical care, and finance, and can also be used for purposes such as electricity bill generation. Therefore, data access control is required to grant access rights to various data users in the IoT environment who need such IoT data. In addition, IoT data contain sensitive information such as personal information, so privacy protection is also essential. Ciphertext-policy attribute-based encryption (CP-ABE) technology has been utilized to address these requirements. Furthermore, system structures applying blockchains with CP-ABE are being studied to prevent bottlenecks and single failures of cloud servers, as well as to support data auditing. However, these systems do not stipulate authentication and key agreement to ensure the security of the data transmission process and data outsourcing. Accordingly, we propose a data access control and key agreement scheme using CP-ABE to ensure data security in a blockchain-based system. In addition, we propose a system that can provide data nonrepudiation, data accountability, and data verification functions by utilizing blockchains. Both formal and informal security verifications are performed to demonstrate the security of the proposed system. We also compare the security, functional aspects, and computational and communication costs of previous systems. Furthermore, we perform cryptographic calculations to analyze the system in practical terms. As a result, our proposed protocol is safer against attacks such as guessing attacks and tracing attacks than other protocols, and can provide mutual authentication and key agreement functions. In addition, the proposed protocol is more efficient than other protocols, so it can be applied to practical IoT environments.

Keywords: IoT data; CP-ABE; data validation; data nonrepudiation, data accountability; security; authentication

Citation: Lee, J.; Kim, M.; Park, K.; Noh, S.; Bisht, A.; Das, A.K.; Park, Y. Blockchain-Based Data Access Control and Key Agreement System in IoT Environment. *Sensors* **2023**, *23*, 5173. https://doi.org/10.3390/s23115173

Academic Editor: Giuseppe Piro

Received: 5 April 2023
Revised: 26 May 2023
Accepted: 26 May 2023
Published: 29 May 2023

1. Introduction

As IoT devices are deployed in various environments such as houses, farms, factories, and grids, the development and spread of smart cities such as smart homes, smart factories, and smart grids continues. As the amount of data generated and collected by IoT devices increases exponentially, it is predicted that the total amount of data generated annually by 2024 will reach 149 ZB [1]. IoT data are used as source data for services related to finance, medical care, artificial intelligence, and electricity bills.

Data access control technology that can provide IoT data to data users (e.g., managers of smart grids and financial institutions) in an appropriate service environment is required to utilize IoT data as source data for various services. To efficiently utilize IoT data and provide them to data users, the gateway collects IoT data, outsources them to a cloud server, and manages the data through the cloud [2,3]. However, the generated IoT data contain sensitive information such as user personal information, so privacy cannot be guaranteed if the data are indiscriminately viewed by institutions using the data. Moreover, data outsourcing also creates security and privacy concerns because it separates data ownership and data management [4]. Therefore, access control for the data users is necessary to protect personal information and provide only data that meet the attributes of the data user that will use the data. To this end, data access control technology using attribute-based encryption (ABE) [5] has recently attracted attention as a promising technology.

In the case of ciphertext-policy ABE (CP-ABE) [6], each original datum is encrypted in relation to the access control structure set in advance by the encryptor. Data users can only decrypt the ciphertext if the set of attributes he or she uses satisfies the ciphertext access structure. IoT data producers need to be able to provide their data only to organizations that want them through the gateway to ensure privacy. Therefore, since the access structure for IoT data must be determined, using CP-ABE is suitable for the IoT environment.

Additionally, if the cloud server manages the computation and communication of most systems, including outsourced data and access control, it is vulnerable to a single point of failure and data management due to centralization issues [7]. In order to solve this problem, research on the decentralization of cloud servers using blockchains has recently been conducted [8,9]. On the other hand, since IoT data are transmitted and received through open channels, malicious attackers can steal the data to perform attacks such as invasions of privacy, data exfiltration, and data abuse. Therefore, to solve these problems, it is necessary to study the application of ABE and blockchain for data privacy provision and access control. In addition, in order to securely store and provide data, gateways, cloud servers, and data users need to verify that they are valid entities through key agreement.

Therefore, in this paper, we suggest a security system that provides authentication while providing access control. We analyze the trends and problems of systems for secure access control and management of data generated in the IoT environment, and present the direction of blockchain-based access control and key agreement to solve these problems.

The main motivations and contributions of this study based on the problems and challenges mentioned above are as follows:

- Unlike existing IoT data access control systems using blockchains, the proposed system guarantees data protection through mutual authentication and key agreement. The detailed method is as follows: The proposed system provides mutual authentication based on bilinear pairing and secure key agreement based on the DBDH assumption. In addition, it provides secure data outsourcing and data access control based on CP-ABE by using the session key generated through key agreement.
- The gateway and the cloud server generate a session key through key agreement and mutual authentication, and the gateway can safely outsource data through the session key. Gateways can also prove data validation through self-signing when uploading data. Data users can request data to the cloud server and verify the received data through the gateway's signature. Thus, the system can provide data accountability.
- Since the proposed system utilizes a public permissioned blockchain, only data users, gateways, and cloud servers registered with the TA (trusted authority) can use the blockchain as a participant. By auditing the blockchain through data users, nonrepudiation of data can be avoided.
- Detailed formal security validation utilizing the widely accepted "AVISPA Software Verification Tool" [10], "indistinguishability game against selective chosen plaintext attack (IND-CPA)", and "informal (nonmathematical) security analysis" shows that the suggested system guarantees safety against multiple potential attacks on smart city environments utilizing IoT.

- Testbed experiments with cryptographic primitives in a laptop environment were performed using the popular "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [11].

The remainder of this paper is organized as follows: Section 2 reviews papers on data access control using CP-ABE and blockchain in IoT environments. Section 3 outlines the proposed system model, blockchain, access structure, bilinear pairing, DBDH assumption, and adversary model. Section 4 describes our proposed data access control system. Section 5 describes the results of formal security validation using AVISPA and IND-CPA, and Section 6 describes the results of informal security analysis. We analyze the efficiency and security features of the protocol in Section 7. Finally, Section 8 concludes the paper.

2. Related Works

Numerous studies on data access control using CP-ABE have been proposed; its application to the IoT environment has also been proposed. In 2007, Ling and Newport [12] proposed a CP-ABE method that can be applied to both positive and negative attributes using an AND gate access structure. They proposed a structure that has been proven to be secure with plaintext selected under the decisional bilinear Diffie–Hellman (DBDH) assumption. Lewko and Waters [13] suggested a CP-ABE method based on multiauthority, and argued that their system does not require collaboration between rapid institutions. However, in the initialization phase, all agencies must set key parameters, so their structure is impractical for large-scale systems.

In order to efficiently store and manage data, systems in which data are outsourced to a cloud server and controlled have been also proposed [14–18]. Yeh et al. [14] proposed a system that can collect patient information from IoT devices and use it for smart healthcare. For data integrity in their system, data are pre-encrypted before uploading to cloud servers, giving patients access control to data. Miao et al. [15] proposed a CP-ABE-based data access control and keyword search system structure in a cloud-enabled mobile crowdsourcing environment. Liu et al. [16] proposed an e-healthcare record system that uploads and shares health data collected from wearable IoT devices to a cloud server and protects the personal information of patients based on CP-ABE. Ding et al. [17] proposed a structure that can ensure data security in IoT systems by using a pairing-free-based CP-ABE in IoT systems. Lu et al. [18] proposed a secure data sharing system in cloud computing that ensures data privacy protection in resource-constrained mobile terminals. However, since these studies are data access control systems based on cloud servers, a centralization problem may occur, which may cause a single-point-of-failure problem.

Therefore, CP-ABE systems have been proposed for access control of IoT data based on blockchains to solve this centralization problem [19–24]. In 2018, Zhang et al. [19] proposed a user-controlled data sharing system with privacy protection using fine-grained access control based on a blockchain and CP-ABE. In 2019, Ding et al. [20] also proposed an ABE access control system for IoT. Blockchain technology was used to record the distribution of properties to prevent single-point errors and data tampering. They demonstrated that authentication can ensure strict access control, but there is no algorithm or protocol for this in their system. Guo et al. [21] suggested a multiauthority blockchain-based ABE protocol for telemedicine systems. Unfortunately, Son et al. [25] figured out that Guo et al.'s protocol [21] is not suitable for real-world environments as patients must maintain their own attribute keys. Yang et al. [22] proposed an EHR sharing system utilizing cloud computing based on ABE and blockchains. In 2021, Wei et al. [23] designed an ABE algorithm for multiauthority scenarios with resource-constrained IoT devices in mind, thereby shifting data management to a blockchain instead of a central server. Qin et al. [24] also proposed a blockchain-based CP-ABE system using cloud computing with consideration to the resource limitations of IoT devices. However, the authentication they proposed [19–24] is authentication for data, not mutual authentication between entities participating in communication. For secure communication, the session key must be calculated by performing mutual authentication and key agreement.

Blockchain-based CP-ABE access control systems have been proposed in various smart environments using IoT devices. However, most studies do not provide mutual authentication, key agreement, data access control, validation and accountability at the same time. Therefore, we propose a structure that guarantees a secure data outsourcing process through mutual authentication and key agreement and provides data access control using CP-ABE technology. In addition, our proposed structure proposes an access control system that provides functions of data nonrepudiation, data accountability, and data validation based on a blockchain.

3. System Models and Preliminary Work

We present the proposed system model for IoT data access control considering data users in the different IoT environments. We describe blockchain characteristics, ABE, and the adversary model used in our system. Table 1 is an explanation of abbreviations and symbols used in this paper.

Table 1. Notations (author's own processing).

Notations & Abbreviations	Meanings
IoT	Internet of Things
ABE	Attribute-based encryption
CP-ABE	Ciphertext-policy ABE
DBDH	Decisional bilinear Diffie-Hellman
DU_i, $DUID_i$, $DUPW_i$	ith data user and his/her identity and password, respectively
GW_j, GID_i	jth gateway and its identity, respectively
CS, ID_{cs}	Cloud server and its identity, respectively
$HDUID_i$, $PGID_j$, PID_{cs}	The hidden identity of data user, gateway, and cloud server, respectively
TA	Trusted authority
R_{cs}, r_i, r_j, k_{TA}	The secret key of CS, DU_i, GW_j, and TA, respectively
PK_{cs}, PK_i, PK_j, PK_{TA}	The public key of CS, DU_i, GW_j, and TA, respectively
$ATTRI_i$	The attribute of DU_i
$attr_i$	The attribute private key of DU_i
$\mathcal{T}$, τ	Access tree and root of tree
SK	The session key established among GW_j and CS
K	The data decryption key
h_1, h_2	Hash function and map-to-point hash function
$\|\|$	Data concatenation operator
$\oplus$	Bitwise exclusive-or operator

3.1. System Model

Our proposed data access control system model is described in Figure 1. The proposed system model consists of the following four entities:

- **Cloud Server (CS)**: A set of CSs forms a "*CS* network", where a distributed ledger is maintained for block additions. CSs are honest but curious entities. Moreover, the *CS* receives the IoT data and provides the data to the data user when the user's attribute value matches. In addition, the *CS* uploads data such as data attributes, signature values, and public keys to the blockchain to solve the centralization problem.
- **Gateway (GW)**: Gateways are distributed in various smart environments that make up the smart city. The gateway collects IoT data from each environment and uploads them to a cloud server with attribute-based encryption appropriate for each attribute.
- **Data User (DU)**: Data user refers to a person who charges fees using IoT data or provides services such as artificial intelligence, finance, and medical care using IoT data. The data user requests an attribute key from the *TA*. After that, the *TA* can request data matching the attribute from the cloud server, and can obtain the original data by decrypting the received data through the attribute key.

- **Trusted Authority (TA)**: All data users, gateways, and cloud servers must register with a fully trusted *TA*.
- **Blockchain**: In the proposed system model, the blockchain is composed of a public permissioned blockchain. To solve the problem of centralization of CSs, the blockchain stores the storage address of data stored in the *CS*, the public key of each component, hash, data access tree, etc., on behalf of the *CS*. The "practical Byzantine fault tolerance (PBFT) consensus algorithm" [26] has been applied for adding blocks to existing blockchains, verifying blocks, and voting-based consensus algorithms. Data users audit the blockchain ledger. All blockchain members can read the ledger, but only data users and cloud servers can upload transitions to the blockchain. When the *DU* requests information from the *CS*, the *CS* checks whether the access tree of the requested information and the attributes of the *DU* match through the blockchain. If the attributes match the access tree, the *CS* passes the encrypted data to the *DU*.

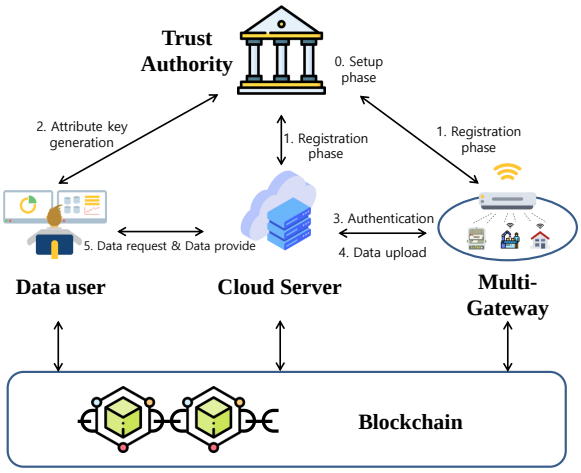

Figure 1. Proposed system model (author's own processing).

In the setup phase, *TA* generates and publishes parameters necessary for the system and tree. During the registration phase, *DU*s, *GW*s, and *CS* are registered with *TA* through closed channels. Through the attribute key generation phase, *DU* can ask *TA* for a key that matches his attribute, and use the acquired attribute key to decrypt the encrypted data. In the authentication phase, *GW* and *CS* perform authentication and key agreement for data upload. In the data upload phase, *GW* uploads data to the cloud server through the agreed session key. Simultaneously, *GW* uploads the signature as a verification value to verify its own data and the upload time to the blockchain. In addition, *CS* uploads the attribute tree value of the data and the record address value where the data are stored to the blockchain. In the data request and provide phase, *DU* requests data from *CS*, and *CS* verifies the *DU*'s request message, checks the attribute value of *DU* through the blockchain, and transmits the corresponding data to *DU*. *DU* downloads the verification value from the blockchain for the transmitted data, verifies that they are valid data, and can decrypt the data with its own attribute key.

3.2. Blockchain

A blockchain is a distributed data storage system that can solve the single-point-of-failure problem that can occur by being concentrated in the cloud server. The decentralized nature of blockchains can provide nonrepudiation of data, accountability, and transparency. In addition, the timestamp recorded on the blockchain allows blockchain participants to know the transaction generation time [27]. In general, four types of blockchain platforms are defined:

- **Public permissionless blockchain**: A public permissionless blockchain provides a 'low trust' environment where anyone can run nodes and participate in the network.

A public permissionless blockchain can be accessed by anyone, and any node can participate in the consensus protocol. Moreover, anyone can read the entire ledger of transactions.

- **Public permissioned blockchain**: Public permissioned blockchains have rules that determine who can participate in the verification process and launch nodes. They are commonly used by public institutions such as government agencies, businesses, or educational institutions. Whitelisted nodes can participate in the consensus mechanism. Owners create validator nodes that define governance rules for the blockchain, including those who can create new nodes or write to the blockchain. However, read access can be used by anyone who makes the blockchain publicly accessible.
- **Private permissionless blockchain**: A private permissionless blockchain has no restrictions on who can participate in the consensus mechanism. However, unlike public permissionless convex chains, there are restrictions on who can read and write content on the blockchain.
- **Private permissioned blockchain**: These blockchains are controlled by a unique group of one or several owners who determine the participants in the consensus mechanism. Only selected user groups can read or write to these blockchains. If public verification of records is not required, consider private permissioned blockchains.

In this paper, only cloud servers and data users of smart cities can write to the blockchain. Therefore, in this paper, a public permission-type blockchain is adopted, and the consensus algorithm uses PBFT.

3.3. Access Structure

According to [6], we use the following access tree as an access structure.

Assuming that $\mathcal{T}$ is an access tree, $\mathcal{T}$ includes $(v, num_v, threshold_v, par(v), ind(v))$, where v is a node of $\mathcal{T}$, $threshold_v$ is threshold value of v, num_v is the number of children nodes of v, $ind(v)$ is unique index of v, and $par(v)$ is a parent node of v. Assuming v is an internal node, v is the threshold gate denoted by AND and OR. AND and OR gates are defined as follows: when $0 < threshold_v \leq num_v$, it is an AND gate if $threshold_v = num_v$ and an OR gate if $threshold_v = 1$.

Moreover, in the case where v is a leaf node, it is described as the attributes $threshold_v = 1$. To fit $\mathcal{T}$ with attribute set $att(v)$, $att(v)$ have to match the threshold gate at root node τ of $\mathcal{T}$. Here, $att(v)$ is defined only if v is a leaf node and represents an attribute related to leaf node v in the tree. In the first case, τ is an attribute and its key satisfies the access tree $att(v)$. In the following case, if τ is a threshold gate whose child node is an attribute, the access tree is satisfied when $att(v)$ holds the threshold gate of τ. In other cases, such as where τ is a threshold gate and the child nodes are also threshold gates, the method in the second case can be applied recursively to solve it.

3.4. Bilinear Pairing

Let G_1 and G_2 be recursive groups with large prime q, and let them be addition and multiplication groups, respectively. Then, a map that satisfies the following conditions can be applied to the bilinear map $e : G_1 \times G_1 \in G_2$.

- **Efficiency**: For all $P, Q \in G_1$, $e(P, Q)$ are able to be computed in polynomial time.
- **Bilinearity**: For all $P, Q \in G_1$, and for all $x, y \in Z_p^*$, $e(xP, yQ)$ is $e(P, Q)^{xy}$.
- **Nondegeneracy**: Existing $P, Q \in G_1$, then $e(P, Q) \neq 1_{G_1}$, where 1_{G_1} is the identifying element of G_1.

3.5. Decisional Bilinear Diffie–Hellman (DBDH) Assumption

Let G_1 be a group of order q; P be a generator of G_1; and $a, b, c, z \in Z_q$ be chosen randomly. The DBDH assumption [28] is that it is difficult for a probabilistic polynomial time

adversary $\mathcal{A}$ to distinguish $(P^a, P^b, P^c, e(P,P)^{abc})$ from $(P^a, P^b, P^c, e(P,P)^z)$. The advantage ε of $\mathcal{A}$ is defined as follows:

$$|Pr[\mathcal{A}(P^a, P^b, P^c, e(P,P)^{abc}) = 1] - Pr[\mathcal{A}(P^a, P^b, P^c, e(P,P)^z) = 1]| \geq \varepsilon \quad (1)$$

If there is no $\mathcal{A}$ can decide whether $e(P,P)^z = e(P,P)^{abc}$, that is deciding whether $z = abc$ or $z \in Z_q$, with a non-negligible advantage, the DBDH assumption holds.

3.6. Adversary Model

We adopt the widely accepted "Dolev–Yao (DY) threat model" [29] for cryptographic analysis of protocol security. A malicious adversary could, according to the assumptions of the DY model, intercept messages sent over public channels. Attackers can also modify, insert, delete, or eavesdrop on hijacked messages.

- An adversary takes full control of transmitted messages sent over an open wireless channel and learns from the messages. The attacker can then modify, remove, or insert a legitimate message.
- In polynomial time, an adversary is able to only guess one value, because guessing more than one value at a time is "computationally infeasible", for example, guessing an ID and password at the same time.
- An adversary can steal or obtain a valid smart card. The adversary can perform a power analysis attack [30,31] on a smart card to steal sensitive information stored on the smart card.

In addition, this paper adopts the assumption of the "CK adversary model" [32], which is a more powerful attack model considering the actual environment. The CK attack model is considered the de facto standard for modeling key exchange protocols. Therefore, in the CK model, for the session key agreed upon between the communicating parties to be secure, the key exchange protocol must minimize the impact of persistent (long-term) or temporary (short-term) secret leaks.

4. Proposed Data Access Control System for IoT Environments

In this section, we propose a method of access control for IoT data, which overcomes the limitations and security pitfalls of previous access control methods. In addition, the proposed protocol guarantees stronger security through authentication in the existing access control method.

4.1. Setup Phase

TA generates public parameters for use in the system's attribute-based encryption and blockchain. The following steps are followed:

Step SP1: TA generates G_1 and G_2 in the same order q, where G_1 is an additive cyclic group and G_2 is a multiplicative cyclic group. Then, TA generates a bilinear map $e : G_1 \times G_1$. TA chooses the secret keys k_{TA} and ζ in Z_q^*, and chooses $P \in G_1$, where P is a generator. Moreover, TA chooses the hash functions $h_1 : \{0,1\}^* \rightarrow Z_q$ and $h_2 : \{0,1\}^* \rightarrow G_1$.

Step SP2: TA computes the public key $PK_{TA} = k_{TA} * P$, a factor of an attribute key $F = \frac{P}{k_{TA}}$ and a factor for decryption $e(P,P)^{\zeta}$. Then, TA publishes $(G_1, G_2, q, e, P, PK_{TA}, F, e(P,P)^{\zeta}, h_1, h_2)$.

4.2. Registration Phase

For key agreement and authentication, GW, of the IoT environment, CS and DU have to register at TA. This phase runs through a secure channel.

4.2.1. Cloud Server Registration Phase

This phase is also executed over the secure channel:

Step CSR1: A cloud server CS picks its identity ID_{cs} and generates a random number c_{cs}. CS computes $PID_{cs} = ID_{cs} \oplus c_{cs}$. Then, CS sends $\langle PID_{cs}, c_{cs} \rangle$ to the trusted authority TA through a closed channel.

Step CSR2: After that, TA stores PID_{cs} in a its secure database. TA computes $R_{cs} = h(k_{ta}||c_{cs})$ as CS's private key. After that, TA sends $\langle R_{cs} \rangle$ to CS over a secure channel.

Step CSR3: CS computes the public key $PK_{cs} = P * R_{cs}$.

4.2.2. Data User Registration Phase

When a new data user DU_i registers with TA, the following steps are followed:

Step UR1: DU_i chooses unique identity and password $DUID_i$ and $DUPW_i$. DU_i generates random nonces IU_i and a_i, where they are in Z_q^*. Then, DU_i computes $HDUID_i = h_1(DUID_i||IU_i)$ and $HDUPW_i = h_1(DUID_i||IU_i||DUPW_i)$. After that, DU_i sends $HDUID_i$, $HDUPW_i \oplus a_i$ to TA via closed channels.

Step UR2: After TA receives the request message, TA computes $TID_i = (HDUID_i * k_{TA}) * PK_{TA}$ and $A_i = TID_i \oplus (HDUPW_i \oplus a_i)$. TA stores $HDUID_i$ with TID_i in a its secure memory and stores A_i in a smart card SC. Then, TA issues SC to DU_i. At the same time, TA sends $\langle h_1(TID_i), HDUID_i \rangle$ to CS via closed channels.

Step UR3: After receiving SC, DU_i computes $Z_i = h_1(DUID_i||DUPW_i) \oplus IU_i$, $B_i = A_i \oplus a_i = TID_i \oplus HDUPW_i$, $C_i = h_1(TID_i||HDUPW_i)$, and $D_i = r_i \oplus HDUPW_i$. Then, DU_i stores Z_i, B_i, C_i and D_i into SC and computes a public key as $PK_i = r_i * P$.

Step UR4: After receiving message, CS computes $MCS_i = h_1(HMID_i||R_{cs})$ and stores MCS_i in its secure database. CS also stores $h_1(TID_i)$ with $HDUID_i$.

4.2.3. Gateway Registration Phase

In this phase, the following steps are performed in the closed channel:

Step GWR1: A gateway GW_j chooses identity GID_j and generates a random nonce b_j. GW computes $PGID_j = GID_j \oplus b_j$. Then, GW generates a public key $PK_j = r_j * P$ and sends $\langle PGID_j \rangle$ to the trusted authority TA via closed channels.

Step GWR2: After that, TA computes $TGID_j = (h_1(PGID_j) * k_{TA}) * PK_{TA}$ and stores $PGID_j$ with $TGID_j$ in a its secure database. Then, TA sends $TGID_j$ to GW_j over a secure channel. At the same time, TA sends $\langle h_1(PGID_j), TGID_j \rangle$ through secure channels.

Step GWR3: CS computes $GCS_j = h_1(h_1(PGID_j||R_{cs}))$ and stores GCS_j in its secure database. CS also stores $TGID_j$ with $h_1(PGID_j)$.

4.3. Attribute Key Generation Phase

In this phase, the data user with attributes $ATTRI_i$ requests the attribute key from the TA and provides the corresponding key.

Step AKG1: DU chooses his/her attributes $ATTRI_i$ and sends it to TA to request the attribute key.

Step AKG2: After that, TA generates random nonces $ra_i, rb_i \in Z_q^*$. In addition, TA computes $At_i = F(\zeta + ra_i)$ for all $s \in ATTRI_i$, and also computes $At_{i_s} = ra_i P + rb_i H(s)$ and $At'_{i_s} = rb_i P$. Then, TA computes attribute keys $attr_i = (At_i, At_{i_s}, At'_{i_s})$. Finally, TA sends attribute keys $attr_i$ to DU_i.

Step AKG3: After receiving attribute keys, DU_i uploads the transaction $Tx_i = (PK_i, ATTRI_i)$ to the blockchain.

4.4. Authentication and Key Agreement Phase

For uploading the IoT data to the cloud server, GW_j and CS authenticate each other. They authenticate each other to secure mutual trust, and later, by establishing the session key SK, GW_j and CS can configure a secure communication channel. The detailed steps involved in this step are shown below and are summarized in Figure 2.

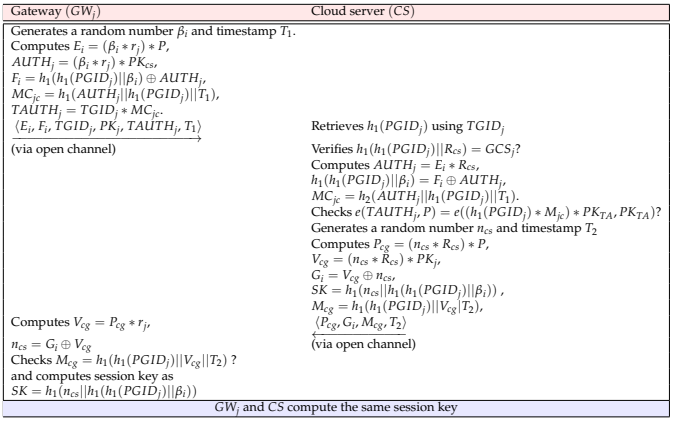

Figure 2. Authentication and key agreement phase (author's own processing).

Step AK1: GW_j generates a random number β_i and timestamp T_1, and computes $E_i = (\beta_i * r_j) * P$, $AUTH_j = (\beta_i * r_j) * PK_{cs}$, $F_i = h_1(PGID_j) \oplus AUTH_j$, $MC_{jc} = h_1(AUTH_j||h_1(PGID_j)||T_1)$, and $TAUTH_j = TGID_j * MC_{jc}$. Then, GW sends a request message $\langle E_i, F_i, PK_j, TAUHT_j, T_1\rangle$ to CS over an insecure channel.

Step AK2: After receiving the message, CS retrieves $h_1(PGID_j)$ using $TGID_j$ and verifies $h_1(h_1(PGID_j)||R_{cs}) \stackrel{?}{=} GCS_j$. If it corrects, CS computes $AUTH_j = E_i * R_{cs}$, $h_1(h_1(PGID_j)||\beta_i) = F_i \oplus AUTH_j$, and $MC_{jc} = h_2(AUTH_j||h_1(PGID_j)||T_1)$. After that, CS checks $e(TAUTH_j, P) \stackrel{?}{=} e((h_1(PGID_j) * M_{jc}) * PK_{TA}, PK_{TA})$. If they are the same, GW_j is authenticated. After that, CS generates a n_{cs} and timestamp T_2. In addition, CS computes $P_{cg} = (n_{cs} * R_{cs}) * P$, $V_{cg} = (n_{cs} * R_{cs}) * PK_j$, and CS also computes $G_i = V_{cg} \oplus n_{cs}$, $SK = h_1(n_{cs}||h_1(h_1(PGID_j)||\beta_i))$ as a session key, and $M_{cg} = h_1(h_1(PGID_j)||V_{cg}||T_2)$. Then, CS sends a response message $\langle P_{cg}, G_i, M_{cg}, T_2\rangle$ to GW_j through public channels.

Step AK3: After that, GW_j checks the validity of $|T_2' - T_2''| < \Delta T$. If it is valid, GW_j computes $V_{cg} = P_{cg} * r_j$ and $n_{cs} = G_i \oplus V_{cg}$. Then, GW_j checks $M_{cg} \stackrel{?}{=} h_1(h_1(PGID_j)||V_{cg}||T_2)$. If it holds, GW_j considers CS as authentic and computes the session key shared with CS as $SK = h_1(n_{cs}||h_1(h_1(PGID_j)||\beta_i))$.

Finally, GW_j and CS complete mutual authentication to generate the same session key SK for IoT data upload.

4.5. Data Upload Phase

GW_j uploads IoT data through the session key agreed with CS. At this time, GW_j encrypts data through CP-ABE and uploads them to CS so that only DU_i with appropriate attributes can access data sharing. In addition, GW_j generates the signature value for data verification of DU_i. CS stores encrypted data and uploads GW_j's signature value, public key, attribute tree, and stored server address value to the blockchain. Detailed steps related to this phase are provided below.

Step DU1: GW_j chooses an access tree $\mathcal{T}$ and root of tree τ. Then, GW_j generates a timestamp TS_j and selects a random polynomial $q_\tau(x)$ with degree $d_\tau = v_\tau - 1$. GW_j generates a random number $x_j = q_\tau(0)$ for a leaf node x of $\mathcal{T}$. Thereafter, GW_j computes $c_{j1} = DATA_j * e(P, P)^{\zeta_{x_j}}$, $c_{j2} = PK_{TA} * s_j$.

For other leaf nodes le of $\mathcal{T}$, GW_j chooses a random point d_{le} of polynomial $q_{le}(x)$. Then, GW_j calculates $C_{le} = P * q_n(0)$ and $C'_{le} = h_2(attr(le)) * q_{le}(0)$ for all leaf nodes le of $\mathcal{T}$. The ciphertext consists of $\delta_j = (\mathcal{T}, c_{j1}, c_{j2}, C_{le}, C'_{le})$. GW_j also computes the signature of data as follows. GW computes $s_j = h_1(PGID_j||r_j||DATA_j)$, $S_j = s_j * P$, and $Sig_j = s_j + h_1(PK_j||\delta_j) * r_j$ as the signature. Finally, GW_j sends $\langle (S_j, Sig_j, \delta_j, TS_j)_{SK}, h_1(PK_j||\delta_j||TS_j)\rangle$ to CS through a open channel.

Step DU2: After that, CS decrypts (Sig_j, δ_j, TS_j) using the session key and checks $h_1(PK_j||\delta_j||TS_j)$. If these values are equal, CS stores δ_j in its database and sets $ADDR_j$ to the

record address. At the end, CS uploads the transaction $Tx_j = (S_j, Sig_j, PK_j, \mathcal{T}, h_1(\delta_j||PK_j), ADDR_j)$ to the blockchain.

4.6. Data Request and Provide Phase

Step DRP1: DU_i inserts the smartcard SC and inputs $DUID_i$ and $DUPW_i$. Then, SC computes $IU_i^{'} = Z_i \oplus h_1(DUID_i||DUPW_i)$, $HDUPW_i^{'} = h_1(DUID_i||IU_i^{'}||DUPW_i)$, and $TID_i^{'} = B_i \oplus HDUPW_i^{'}$. SC checks $C_i^{'} \stackrel{?}{=} h_1(TID_i^{'}||HDUPW_i^{'})$. If it is valid, DU_i generates random nonce r_{du} and timestamp TS_i, and computes $r_i = D_i \oplus HDUPW_i^{'}$, $MC_{ic} = h_1(AUTH_i||h_1(DUID_i||IU_i^{'})||TS_i)$, $AID_i = TID_i * MC_{ic}$. After that, DU_i obtains the transaction $(Sig_j, PK_j, \mathcal{T}, h_1(\delta_j||PK_j), ADDR_j)$. DU_i computes $M_1 = (PK_i||ADDR_j||r_{du}||TS_i) + r_i * PK_{cs}$ and sends the data request message $\langle h_1(TID_i)\ AID_i, PK_i, TS_i, M_1\rangle$.

Step DRP2: After receiving the message, CS retrieves $HDUID_i$ using $h_1(TID_i)$ and verifies $h_1(HDUID_i^{'}||R_{cs}) \stackrel{?}{=} MCS_i$. If it holds, CS computes $AUTH_i = PK_i * R_{cs}$ and $MC_{ic} = h_1(AUTH_i||HDUID_i||TS_i)$. Then, CS checks $e(AID_i, P) \stackrel{?}{=} e((HDUID_i * MC_{ic}) * PK_{TA}), PK_{TA})$. If this equality holds, CS obtains $(PK_i, ATTRI_i)$ from the blockchain. Then, CS computes $(PK_i||ADDR_j||r_{du}||TS_i) = M_1 - R_{cs} * PK_i$ and confirms that $ATTRI_i$ satisfies tree of δ_j. If it is met, CS calculates $M_2 = (\delta_j||T_{cs}) + R_{cs} * PK_i$. Then, CS sends the message $\langle M_2\ AID_i, TS_{cs}\rangle$.

Step DRP3: After receiving the message, DU_i computes $(\delta_j||TS_{cs}) = M_2 - r_i * PK_{cs}$. Then, DU_i checks $h_1(\delta_j||PK_j)^* \stackrel{?}{=} h_1(\delta_j||PK_j)$ acquired on the blockchain. Depending on the type of root node, data decryption proceeds as follows.

- Case 1: If τ is a leaf node, DU_i calculates $e(A_{t_i}, C_{le})$ and $e(A_{t_i}^{'}, C_{le}^{'})$. Then, DU_i computes At_{i_s}, C_{le} and $At_{i_s}^{'}, C_{le}^{'}$. Then, DU_i computes $\frac{e(At_{i_s}, C_{le})}{e(At_{i_s}^{'}, C_{le}^{'})} = e(P,P)^{ra_i q_\tau(0)} = K$ for data decryption. Thereafter, DU_i can decrypt as follows:

$$\begin{aligned}\frac{c_{j1}}{e(c_{j2}, At_i)/K} &= \frac{DATA_j * e(P,P)^{\zeta x_i}}{e(x_i PK_{TA}, F(\zeta + r_{ai}))/K} \\ &= \frac{DATA_j * e(P,P)^{\zeta x_i}}{e(P,P)^{x_i(\zeta + r_{ai})}/K} = DATA_j\end{aligned}$$

- Case 2: We assume that root node τ is a threshold gate and child nodes are attributes. Before we describe the decryption computation, we define the symbols c_τ and $\Delta_{ind(le),c_\tau}(x)$. c_τ is a set of child nodes of the root node, and $\Delta_{ind(le),c_\tau}(x)$ is Lagrange coefficient, where $\Delta_{ind(le),c_\tau}(x) = \Pi_{o \in c_\tau, ind(o) \neq ind(le)} \frac{x - ind(o)}{ind(le) - ind(o)}$.
  DU_i computes $\frac{e(At_{i_s}, C_{le})}{e(At_{i_s}^{'}, C_{le}^{'})} = e(P,P)^{ra_i q_\tau(0)} = K_{le}$ for all leaf nodes le. After that, DU_i computes decrypt key:

$$\begin{aligned}\prod_{le} K_{le}^{\triangle_{ind(le),c_\tau(0)}} &= \prod_{le} (e(P,P)^{ra_i q_{le}(0)})^{\triangle_{ind(le),c_\tau(0)}} \\ &= \prod_{le} (e(P,P)^{ra_i q_\tau(ind(le))})^{\triangle_{ind(le),c_\tau(0)}} \\ &= e(P,P)^{ra_i q_\tau(0)} = K\end{aligned}$$

  Then, DU_i can decrypt the IoT data.

4.7. Data Validation Phase

If the data users want to verify that the gateway information is correct, data verification can be performed during this phase. This data validation ensures that the gateway is accountable for its own data and that the data user can obtain the reliability of the data. A detailed description of this phase is provided bellow:

Step DVP: DU_i obtains S_j, Sig_j, PK_j, and $h_1(\delta_j||PK_j)$ from the transaction related to the data. DU_i computes $Sig_j * P = s_j * P + h_1(PK_j||\delta_j) * r_j * P = S_j + h_1(PK_j||\delta_j) * PK_j$. Then, DU_i checks $S_j = Sig_j - h_1(PK_j||\delta_j) * PK_j$. If it is valid, DU_i can be considered as data validation completed.

4.8. Block Formation and Addition Phase

In the key generation phase and data upload phase, DU_i and CS create a transaction and upload it to the blockchain. We describe it in detail in terms of CS in this section, and the block construction and addition of DU_i is similar. The "practical Byzantine fault tolerance (PBFT) consensus algorithm" [26] has been applied for adding blocks to existing blockchains, verifying blocks, and voting-based consensus algorithms. The block structure is depicted in Figure 3, and the entire algorithm of block addition is given in Algorithm 1.

Block Header	
Block version (BV)	Unique block version number
Timestamp	Block creation time
Merkle tree root (MR)	Merkle tree root on the list of all transactions
Previous block hash $PBHash = (Block_{m-1})$	Hash value of the previous block
Owner of block	CS
Public key of the signer	PK_{cs}
Block Payload	
List of transactions (Tx_j)	$\{TX_j \mid j = 1, 2, ..., t\}$
Current hash ($CBHash$)	Hash of Block header
Signature on transactions	$ECDSA.sig_{Tx}$

Figure 3. Formation of a block on the transactions by CS (author's own processing).

Algorithm 1 PBFT Consensus for Block Addition in Blockchain by Cloud Server

1: **Input:** Block ($Block_m$ as shown in Figure 3, transactions pool (Tx_pol), transactions threshold ($Tx_{thresh=t}$, number of CS nodes : n_{cs}, minimal approval ($Min_{approve} = 2 * (n_{cs} - 1)/3 + 1$)
2: **Output:** Commitment for block addition (CMP)
3: Assume that a cloud server node (CS_l) is elected as a leader
4: CS_l picks a fresh timestamp and creates a block $Block_m$ with Tx_{pool}
5: CS_l sets $CMP = NULL$ and sends $Block_m$ to follower cloud server nodes ($CS_k(k \neq l|k = 1, 2, \ldots, n_{cs})$) for voting request
6: **for** each follower CS_j **do**
7: **if** (($Tx_j = valid$) and ($MR = valid$) and ($ECDSA.sig_{Tx} = valid$) and ($CBHash = valid$)) **then**
8: Set $CMP = CMP + 1$
9: **end if**
10: **end for**
11: **if** ($CMP \geq Min_{approve}$) **then**
12: Add $Block_m$ to the blockchain
13: Broadcast commitment message to CS
14: **end if**

4.8.1. Block Formation Phase

At the data upload phase of our system, the data generated by GW_j are uploaded to CS using SK agreed between GW_j and CS at the authentication and key agreement phase. CS safely gathers t counts of data, filters that information, and then generates t counts of transactions $Tx_j = (S_j, Sig_j, PK_j, \mathcal{T}, h_1(\delta_j||PK_j), ADDR_j)$, for j = 1, 2, ..., t, to contribute to the transactions pool. To describe this in detail in terms of the data upload phase, CS computes the Merkle tree root (MR) for transactions Tx_j and calculates "elliptic curve

digital signature" for transactions Tx_j as $ECDSA.sig_{Tx} = ECDSA.sig_{gen}(Tx_{msg})$, where $Tx_{msg} = h_1(Tx_1||Tx_2||...||Tx_j||PK_{cs}||MR)$.

4.8.2. Block Addition Phase

After block formation phase, the MR for the transaction existing in the block is verified. In addition, CS conducts a voting-based PBFT consensus algorithm. The CS nodes $CS_l|l = 1, 2, ..., n_{cs}$ (n_{cs} represent the number of peers in CS) form a distributed P2P network. Here, each CS node is considered a peer node that is responsible for adding blocks. After the CS peer node receives the $Block_m$, peer node verifies it with the existing transaction pool. When all transactions in $Block_m$ are confirmed by the transaction pool, the peer puts a valid vote into the commit message pool. CS constantly checks the commit message pool and checks when the minimum approval $(Min_{approve})$ for block additions on the blockchain is reached; where $Min_{approve} = 2 * (n_{cs} - 1)/3 + 1$, the new block $Block_m$ will be added to the blockchain.

5. Formal Security Validation: AVISPA Simulation Study and IND-CPA

In this section, we utilize the "AVISPA simulation tool" [10] and IND-CPA to verify the security of the proposed system.

5.1. AVISPA Simulation

We use the "AVISPA Simulation Tool" [10] in this section to validate our proposed system security against man-in-the-middle and replay attacks.

In AVISPA, there are four backends: "tree automata based on automatic approximations for analysis of security protocols (TA4SP)", the "SAT-based model checker (SATMC)", the "on-the-fly-mode-checker (OFMC)" and the "constraint-logic-based attack Searcher" (CL-AtSe)". Among these, the SATMC and TA4SP backends can not aid the "bitwise exclusive OR (XOR)". However, since our system has an XOR operation, two backends are not suitable for analysis. Therefore, we adopt two backends, OFMC and CL-AtSe, which support XOR operation, and use them for analysis. In the proposed system, "High-Level Protocol Specification Language (HLPSL)", a language supported by AVISPA, is used to implement the basic roles of CS and GW_j. Figure 4 shows the HLPSL implementation of the role user.

At transition 1, GW sends the request message $\{PGID_j\}$ to TA using SND operation and $SKgwta$, which means the secure channel. The declaration $secret(\{Bj^{'}, Rj\}, sp3, \{GW\})$ means that the random nonce B_j and secret key R_j are only known to GW.

At transition 2, GW receives the $TGID_j$ from TA. In login and authentication phase, GW sends the message $\{E_i, F_i, TGID_j, PK_j, TAUTH_j, T_1\}$ to CS through insecure channel. The declaration $witness(GW, CS, gw\_cs\_bei, Bei')$ means that GW generates a random nonce β_i for CS.

At transition 3, GW receives the message $\{P_{cg}, G_i, M_{cg}, T_2\}$ from CS. The declaration $request(CS, GW, cs\_gw\_ncs, Ncs')$ specifies that CS request to the GW for checking the value of n_{cs}.

HLPSL of cloud server is implemented similarly to gateway's HLPSL. In addition, it implements "composite roles and goals for sessions and environment" of the proposed system through HLPSL. AVISPA used in this section is a security validation simulation based on the DY model [30]. Figure 5 gives the analysis results performed on the CL-ATse and OFMC backends. The figure clearly shows that the proposed system can be resistant to "replay and man-in-the-middle attacks".

```
%%%%% role GW %%%%%
role gatew(CS,GW,TA:agent, SKcsta,SKgwta:symmetric_key, H,Mul:hash_func, SND,RCV:channel(dy))
played_by GW
def=
local State: nat,
P,GIDj,PGIDj,Xta,PKta,PKj,Bj,Rj,Ccs,Bei,T1,Ei,AUTHj,Fi,MCjc,TAUTHj,Ncs,T2,SK:text
const sp3,gw_cs_bei,cs_gw_ncs: protocol_id
init State:=0
transition

1. State = 0 /\ RCV(start) =|>
State':=1
/\ Bj':=new()
/\ PGIDj':=xor(GIDj,Bj')
/\ PKj':=Mul(Rj.P)
/\ SND({PGIDj'}_SKgwta)
/\ secret({Bj',Rj},sp3,{GW})

2. State = 1 /\ RCV({Mul(H(xor(GIDj,Bj')).Xta.PKta)}_SKgwta) /\ RCV(Mul(P.H(Xta.Ccs'))) =|>
State':=2
/\ Bei':=new() /\ T1':=new()
/\ Ei':=Mul(Bei'.Rj.P)
/\ AUTHj':=Mul(Bei'.Rj.Mul(P.H(Xta.Ccs')))
/\ Fi':=xor(H(H(xor(GIDj,Bj')).Bei'),AUTHj')
/\ MCjc':=H(AUTHj'.H(xor(GIDj,Bj'))).T1')
/\ TAUTHj':=Mul(Mul(H(xor(GIDj,Bj')).Xta.PKta).MCjc')
/\ SND(Ei'.Fi'.Mul(H(xor(GIDj,Bj')).Xta.PKta).Mul(Rj.P).TAUTHj'.T1')
/\ witness(GW,CS,gw_cs_bei,Bei')

3. State = 2 /\ RCV(Mul(Ncs'.H(Xta.Ccs').P).xor(Mul(Ncs'.H(Xta.Ccs').Mul(Rj.P)),H(H(xor(GIDj,Bj')).Mul(Ncs'.H(Xta.Ccs').Mul(Rj.P)).T2')).H(H(xor(GIDj,Bj')).
Mul(Ncs'.H(Xta.Ccs').Mul(Rj.P)).T2').T2') =|>
State':=3
/\ SK':=H(Ncs'.H(H(xor(GIDj,Bj')).Bei))
/\ request(CS,GW,cs_gw_ncs,Ncs')

end role
```

Figure 4. HLPSL specification for user (Author's own processing).

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS

BOUNDED_NUMBER_OF_SESSIONS

PROTOCOL

/home/span/span/testsuite/results/IoT_Lee.if

GOAL
  as_specified

BACKEND
  OFMC
COMMENTS
STATISTICS

  parseTime: 0.00s
  searchTime: 1.21s
  visitedNodes: 1040 nodes
  depth: 9 plies
```

```
SUMMARY
  SAFE

DETAILS

BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL

/home/span/span/testsuite/results/IoT_Lee.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 15 states
  Reachable  : 15 states
  Translation: 0.02 seconds
  Computation: 0.00 seconds
```

Figure 5. Simulation results on OFMC and CL-AtSe.

5.2. IND-CPA Security

We prove the confidentiality property of our system with the game of IND-CPA. In our scheme, the game is defined as follows.

- **Init.** The adversary $\mathcal{A}$ gives a challenge access structure $\mathcal{T}^*$.
- **Setup.** The simulator $\mathcal{X}$ executes Setup phase and sends the public parameters to the adversary $\mathcal{A}$.

- **Phase 1.** $\mathcal{A}$ queries multiple private keys corresponding to q_1 different sets of attributes $(ATTRI_1, ..., ATTRI_{q_1})$ where $ATTRI_i \notin \mathcal{T}^*$.
- **Challenge.** $\mathcal{A}$ submits two plaintext $DATA_0$ and $DATA_1$, where $|DATA_0| = |DATA_1|$ to the simulator $\mathcal{X}$ with $\mathcal{T}^*$. $\mathcal{X}$ flips the coin $b \in \{0,1\}$, encrypts $DATA_b$ under $\mathcal{T}^*$, and sends the ciphertext CT^* to $\mathcal{A}$.
- **Phase 2.** $\mathcal{A}$ repeats Phase 1 with the attribute sets $(ATTRI_{q_1+1}, ..., ATTRI_q)$ where $ATTRI_i \notin \mathcal{T}^*$.
- **Guess.** $\mathcal{A}$ outputs a guess b' of b to the simulator $\mathcal{X}$. If $b' = b$, $\mathcal{A}$ wins the game.

The adversary $\mathcal{A}$'s advantage ε in this game is defined as $\varepsilon = |Pr[b' = b] - \frac{1}{2}|$. If $\mathcal{A}$ in probabilistic polynomial time can be played with a non-negligible advantage ε, then we prove that the problem of the DBDH assumption can be solved with $\varepsilon/2$.

Proof. Assume that the adversary $\mathcal{A}$ wants to take advantage of ε to subvert the system. We build a $\mathcal{X}$ simulator to play the DBDH game with a $\varepsilon/2$ advantage. We proceed through the simulation process as follows. The $\mathcal{B}$ challenger randomly picks $a, b, c, z \in Z_q$ and generator $P \in G_1$. $\mathcal{B}$ flips a coin to obtain a random value $\mu \in \{0,1\}$. If $\mu = 1$, $Z = e(P,P)^z$, which means $(P^a, P^b, P^c, e(P,P)^z)$. Otherwise, $Z = e(P,P)^{abc}$ means $(P^a, P^b, P^c, e(P,P)^{abc})$. After that, $\mathcal{B}$ transmits the results to $\mathcal{X}$.

Init. The simulator $\mathcal{X}$ runs $\mathcal{A}$ to create access structure $\mathcal{T}^*$ that $\mathcal{A}$ hopes to attack. Then, $\mathcal{A}$ transmits it to $\mathcal{X}$.

Setup. $\mathcal{X}$ computes public parameters $\{PK_{TA} = k_{TA} * P, F = \frac{P}{k_{TA}}, e(P,P)^{\zeta}\}$, where $\zeta = ab$. Then, $\mathcal{X}$ sends them to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ requests multiple private keys $(attr_{i1}, ..., attr_{iq_1})$ corresponding to q_1 different sets of attributes $(ATTRI_1, ..., ATTRI_{q_1})$ where $ATTRI_i \notin \mathcal{T}^*$. The simulator $\mathcal{X}$ generates random nonces $ra_i, rb_i \in Z_q^*$. $\mathcal{X}$ computes $At_i = F(\zeta + ra_i)$ for all $s \in ATTRI_i$, $AT_{i_s} = ra_i P + rb_i H(s)$, $At'_{i_s} = rb_i P$, $attr_i = (At_i, At_{i_s}, At'_{i_s})$. Then, $\mathcal{X}$ sends $attr_i$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ submits $\mathcal{T}^*$ to the $\mathcal{X}$ simulator with plain text $DATA_0$ and $DATA_1$ of equal length. $\mathcal{X}$ randomly tosses a coin to obtain $b \in \{0,1\}$. If $\mu = 0$, then $Z = e(P,P)^{abc}$. In this case, we let $x_j = c$, then $e(P,P)^{abc} = e(P,P)^{\zeta x_j}$ and $c_{j1} = DATA_b * e(P,P)^{abc}$. Otherwise, if $\mu = 1$, then $Z = e(P,P)^z$ and $c_{j1} = DATA_b * e(P,P)^z$. $\mathcal{X}$ computes $c_{j2} = PK_{TA} * s_j$. Then, $\mathcal{X}$ chooses a random point d_{l_e} of polynomial $q_{le}(x)$ and computes $C_{le} = P * q_n(0)$, $C'_{le} = h_2(attr(le)) * q_{le}(0)$ for all leaf nodes le of $\mathcal{T}$. Then, $\mathcal{X}$ sends $\delta_j = (c_{j1}, c_{j2}, C_{le}, C'_{le})$ to $\mathcal{A}$.

Phase 2. The adversary $\mathcal{A}$ repeats **Phase 1** to obtain the private keys that are associated with attribute sets $\forall ATTRI_i|_{q_1+1 \le i \le q}$ and $ATTRI_i \notin \mathcal{T}^*$.

Guess. $\mathcal{A}$ guesses b' of b. If $b \neq b'$, $\mathcal{X}$ gives a result 1, otherwise, it gives a result 0. If $\mathcal{X}$ gives a result 0, then $Z = e(P,P)^{abc}$. $\mathcal{A}$ can obtain practical ciphertext δ_j. The advantage in this case is ε, so we obtain $Pr[b' = b|Z = e(P,P)^{abc}] = \frac{1}{2} + \varepsilon$. When $\mathcal{X}$ gives a result 0, it means $Z = e(P,P)^z$. $\mathcal{A}$ obtains the wrong ciphertext, and does not have the advantage of guessing the correct b', so it is able to obtain $Pr[b \neq b|Z = e(P,P)^z] = \frac{1}{2}$. Therefore, the probability Pr of a successful game is

$$\begin{aligned} Pr &= \frac{1}{2} Pr[\mathcal{A}(P, P^a, P^b, P^c, e(P,P)^{abc}) = 1] \\ &\quad + \frac{1}{2} Pr[\mathcal{A}(P, P^a, P^b, P^c, e(P,P)^z) = 1] - \frac{1}{2} \\ &= \frac{1}{2} Pr[b' = b|Z = (P,P)^{abc}] \\ &\quad + \frac{1}{2} Pr[b' \neq b|Z = e(P,P)^z] - \frac{1}{2} \\ &= \frac{1}{2} \times (\frac{1}{2} + \varepsilon) + \frac{1}{2} \times \frac{1}{2} - \frac{1}{2} \\ &= \frac{\varepsilon}{2} \end{aligned} \tag{2}$$

Therefore, our scheme ensures IND-CPA security.
□

6. Informal Security Analysis

We provide an nonmathematical (informal) security analysis of whether the proposed system can provide various security features and safety against possible attacks.

6.1. Correctness of Data Decryption Key

If the leaf node le is a root node τ, we check the correctness of the decryption key as follows:

$$\begin{aligned}\frac{e(At_{i_s}, C_{le})}{e(At_{i_s'}, C'_{le})} &= \frac{e(ra_iP + rb_ih_2(attr(\tau)), q_\tau(0)P)}{e(rb_iP, h_2(attr(\tau))q_\tau(0))} \\ &= \frac{e(ra_iP, q_\tau(0)P)e(rb_ih_2(attr(\tau)), q_\tau(0)P)}{e(rb_iP, h_2(attr(\tau))q_\tau(0))} \\ &= \frac{e(P,P)^{ra_iq_\tau(0)}e(h_2(attr(\tau)), P)^{rb_iq_\tau(0)}}{e(P, h_2(attr(\tau)))^{rb_iq_\tau(0)}} \\ &= e(P,P)^{ra_iq_\tau(0)} = K\end{aligned}$$

6.2. Guessing Attacks

The malicious adversary $\mathcal{A}$ cannot guess the data user's $DUID_i$ and $DUPW_i$ in the proposed system. $\mathcal{A}$ obtains the credentials $\{Z_i, B_i, C_i, D_i\}$ stored on the smart card. However, since $\{Z_i, B_i, C_i\}$ is encrypted with random numbers IU_i and a_i, $\mathcal{A}$ cannot obtain sensitive information. Furthermore, these values are protected via "a one-way collision-free hash function $h(\cdot)$". In addition, D_i is masked by the unknown parameter $HDUPW_i$ and secret key r_i. As a result, our proposed system can resist guessing attacks.

6.3. Tracing Attacks and Provides Anonymity

The adversary $\mathcal{A}$ is trying to obtain the real IDs of DU_i and GW_j to perform a tracking attack. In our system, the user's real identity $DUID_i$ is hidden by $HDUID_i$ masked with a random number IU_i. In addition, the DU_i sends the message through the public channel using the temporary ID TID_i received from TA via an insecure channel. Moreover, GW_j hides its real ID GID_j as $PGID_j$. GW_j sends a message through the public channel with the temporary ID $TGID_j$ obtained from TA. So, $\mathcal{A}$ cannot know original IDs $DUID_i$ and GID_j. This demonstrates that our system provides anonymity and can resist tracing attacks.

6.4. Impersonation Attacks

$\mathcal{A}$ may attempt to impersonate each entity by calculating legitimate messages to obtain information. In our system, messages sent over public channels are encrypted using random numbers β_i, n_{cs}, x_i, and r_{du} and secret values r_j and R_{cs}. Moreover, in the data upload phase, the message is encrypted by the session key SK. $\mathcal{A}$ tries to take out these values, but this cannot be carried out. In addition, each of the entities check $e(TAUTH_j, P) \stackrel{?}{=} e((h_1(PGID_j) * M_{jc}) * PK_{TA}, PK_{TA})$, $M_{cg} \stackrel{?}{=} h_1(h_1(PGID_j)||V_{cg}||T_2)$, and $e(AID_i, P) \stackrel{?}{=} e((HMID_i * MC_{ic}) * PK_{TA}), PK_{TA})$. Therefore, the proposed system can provide protection against impersonation attacks.

6.5. Ephemeral Secret Leakage Attacks

In the authentication and key agreement phase, GW_j and CS establish the session key $SK = h_1(n_{cs}||h_1(h_1(PGID_j)||\beta_i)) = h_1(n_{cs}||h_1(h_1(GID_j \oplus b_j)||\beta_i))$ in our system. The SK depends on "ephemeral secrets n_{cs} and β_i" and long-term secret b_j. Even if the attacker "short-term secret n_{cs} and β_i" is compromised for $\mathcal{A}$, guessing SK without long-term secret b_j is "a computationally difficult problem." Likewise, even if "long-term secret b_j" is compromised to $\mathcal{A}$, deriving SK is also "computationally difficult. except for short-term

secrets. Since SK between the gateway and the cloud server is distinct and unique, leaking SK from a session to $\mathcal{A}$ is "computationally infeasible" as it applies both short-term and long-term secrets without having to compute another session key in another session. Therefore, the proposed system prevents ephemeral secret leakage attacks.

6.6. Mutual Authentication and Key Agreement

At our system, GW and CS use the $TAUTH_j$ and M_{cg} values to authenticate each other by verifying the message. Every transmitted message is changed with a random number and current timestamps. GW and CS authenticate each other through an authentication and key agreement phase and compute the same session key SK only if the authentication is complete. Therefore, our system provides key agreement through mutual authentication.

6.7. Data Access Control, Validation and Accountability

The proposed system can provide access control to IoT data of GW_j. GW_j establishes an access tree for IoT data and uses it to encrypt data and upload them to CS. Then, only DU_i with the appropriate set of attributes in the IoT data's access tree is able to request data from CS and decrypt them with the attribute key. In addition, GW_j uploads the signature value of its own data to the transaction on the blockchain. DU_i can confirm that the data are uploaded by GW_j through the signature value of the transaction, which means that GW_j guarantees accountability for its own data when uploading. Thus, the system can provide data access control, validation, and accountability.

7. Efficiency Features And Security Analysis

The proposed system is compared with existing competitive data access control systems in the smart city area, such as smart health and smart homes [18,24]. The compared schemes are all schemes using attribute-based encryption. We compare different data access control schemes with each other in terms of communication and communication costs, function, and security features.

7.1. Testbed Experiment Using MIRACL

In this section, we apply MIRACL to show an an environment for practical perspective experiments. The MIRACL testbed experiment shows the computation costs of the proposed system. We performed a testbed experiment with cryptographic primitives using the popular "MIRACL" [11] in a laptop environment. Here are the detailed performance details of the laptops we used: "Ubuntu 18.04.4 LTS with memory 8 GiB, processor: Intel Core i7-4790 @ 3.60 GHz $\times$ 4, CPU Architecture: 64-bit". The experiments were run 100 times to determine the time to run "bilinear pairing operation (T_{pair})", "ECC signature operation (T_{sig}), "ECC scalar point multiplication (T_{mul})", "ECC point addition (T_{add})", "modular exponentiation operation (T_{exp})", "map-to-point-hash-function (T_{mtp})", "encryption function (T_{enc})", "decryption function (T_{dec})", and "one-way-hash-function (T_h)". Thereafter, the average execution time in milliseconds for these functions or operations over 100 run was recorded: 6.587 ms, 0.546 ms, 2.547 ms, 0.013 ms, 0.164 ms, 7.564 ms, 0.001 ms, 0.001 ms, and 0.003 ms, respectively.

7.2. Security and Function Feature Comparison

This section presents the results of comparison of the proposed system with related existing approaches in terms of security and functionality. Table 2 presents the results of the comparison. Previous studies do not provide data accountability, nor do they provide the functions of mutual authentication and key agreement, whereas the proposed method meets all essential security and functional requirements for data access control in a smart city environment.

Table 2. Security and function properties comparison (Author's own processing).

Security and Function Properties	Lu et al. [18]	Qin et al. [24]	Proposed
SF_1	x	-	o
SF_2	x	-	o
SF_3	o	o	o
SF_4	x	o	o
SF_5	-	-	o
SF_6	-	-	o
SF_7	x	x	o
SF_8	o	o	o
SF_9	x	x	o

o: provide the security property x: does not provide the security property -: does not consider SF_1: Guessing attack SF_2: Anonymity and tracing attacks SF_3: Replay and man-in-the-middle attacks SF_4: Impersonation attack SF_5: ESL attack SF_6: Session key disclosure attack SF_7: Mutual authentication and key agreement SF_8: Data validation SF_9: Data accountability.

7.3. Computation Cost Comparison Analysis

Computational costs are compared, taking into account the data upload and data request and provide phases, and follow the testbed experiment results reported in Section 7.1.

We use the average time required on the platform for the data owner/gateway/IoT device, cloud server, and data user costs, respectively. Table 3 shows the comparison results of the computation costs. In Table 3, n means the number of attributes. We assumed that n is 5 to obtain the total computation costs. It can be observed that the total computational costs of our system are slightly higher than those of the other systems. The proposed system uses traditional CP-ABE, which has proven safety rather than efficiency. Moreover, as shown in Table 2, the proposed system can provide mutual authentication, key agreement, and data accountability that other systems cannot provide, and it is safe against attacks from various security aspects.

Table 3. Computation costs comparison (Author's own processing).

System	Data Owner /Gateway	Cloud Server	Data User	Total Costs
Lu et al. [18]	$(1+n)T_{mul}+2T_{exp}+T_h$ $\approx$ 15.613 ms	$T_h+T_{sig}+(n)T_{exp}$ $+(3+n)T_{mul}+$ $(2n)T_{pair}\approx$ 83.013 ms	$T_{pair}+2T_{mul}$ $\approx$ 11.681 ms	110.307 ms
Qin et al. [24]	$5T_{mul}+9T_{exp}+T_{add}+4T_h$ $\approx$ 14.236 ms	$(8+6n)T_{mul}+6T_{pair}$ $\approx$ 137.308 ms	$T_{exp}+2T_{mul}$ $\approx$ 5.258 ms	156.802 ms
Proposed	$(4+2n)T_{mul}+(n)T_{mtp}+$ $2T_h+T_{pair}$ $+T_{add}+T_{enc}\approx$ 80.081 ms	$3T_h+2T_{mul}+2T_{add}+T_{pair}$ $\approx$ 11.716 ms	$5T_{mul}+(n)T_{pair}+(n)T_{exp}$ $+6T_h\approx$ 46.508 ms	138.305 ms

n: number of attribute (assuming that n = 5).

7.4. Communication Cost Comparison Analysis

For comparison analysis of the communication costs during the data upload and data request and provide phases between the proposed system and other systems, the l column matrix, encryption data, hash function output value (using SHA-256), public key, identity, ECC value, chain code, index, and timestamp are taken as 32l bits, 256 bits, 256 bits, 256 bits, 160 bits, 256 bits, 256 bits, 256 bits, and 32 bits, respectively.

Table 4 indicates that our system requires communication costs of 2112 bits to exchange three messages for data upload and data download. On the other hand, the schemes of Lu et al. [18] and Qin et al. [24] require communication costs of 32l+ 1952 bits for three messages and 2208 bits for three messages.

Table 4. Communication costs comparison (Author's own processing).

System	Number of Messages	Number of Bits
Lu et al. [18]	3	$32l + 1952$
Qin et al. [24]	3	2208
Proposed	3	2112

8. Conclusions

In this paper, we proposed an access control system for IoT data in various IoT environments based on CP-ABE and blockchains. Existing systems do not provide mutual authentication and key agreement for secure communication. However, the proposed system guarantees secure communication through these two properties. In addition, the proposed system can provide data validation and accountability to data users. To verify the safety of our system, formal and unofficial security analysis was performed, and the proposed system was compared and analyzed with existing systems in terms of security and functionality. Through the analysis results, it was found that the proposed system is safe against guessing, tracing, ESL, and session key disclosure attacks, unlike existing systems. In addition, our protocol can be said to be an efficient protocol because it has a computation cost similar to or lower than that of existing systems and a lower communication cost than existing systems.

In the future, we plan to design a more efficient access control system. In this paper, we used the traditional CP-ABE, but we need to design an efficient ABE for a more efficient system design. In traditional CP-ABE, when the number of users or the number of attributes increase, the number of pairing operations increases. This will increase the computational cost of the system, which will make it impossible to provide real-time services to users in the IoT environment. In order to solve this problem, there is a need to study a new method of access control in the future. If we develop an efficient access control method even if the number of users and attributes increases, we will be able to design an access control system that is more suitable for the IoT environment.

Author Contributions: Conceptualization, J.L., K.P. and Y.P.; software, A.B. and A.K.D.; verification, A.B. and A.K.D.; validation, M.K., K.P. and S.N.; formal analysis, J.L. and M.K.; investigation, K.P. and S.N.; writing—original draft preparation, J.L.; writing—review and editing, S.N., A.K.D. and Y.P.; supervision, Y.P.; funding acquisition, K.P. and S.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Electronics and Telecommunications Research Institute(ETRI) grant funded by the Korean government. [23ZR1330, Core Technology Research on Trust Data Connectome].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Holst, A. Volume of Data/Information Created, Captured, Copied, and Consumed Worldwide from 2010 to 2024. *Statista*. 2020. Available online: https://www.statista.com/statistics/871513/worldwide-data-created/ (accessed on 30 January 2021).
2. Juliadotter, N.V.; Choo, K.K.R. Cloud attack and risk assessment taxonomy. *IEEE Cloud Comput.* **2015**, 2, 14–20. [CrossRef]
3. Osanaiye, O.; Choo, K.K.R.; Dlodlo, M. Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework. *J. Netw. Comput. Appl.* **2016**, *67*, 147–165. [CrossRef]
4. Chen, X.; Li, J.; Huang, X.; Ma, J.; Lou, W. New publicly verifiable databases with efficient updates. *IEEE Trans. Dependable Secur. Comput.* **2015**, *125*, 546–556. [CrossRef]
5. Sahai, A.; Waters, B. Fuzzy identity-based encryption. In Proceedings of the Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; pp. 457–473.
6. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-policy attribute based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.

7. Xie, S.; Zheng, Z.; Chen, W.; Wu, J.; Dai, H.-N.; Imran, M. Blockchain for cloud exchange: A survey. *Comput. Electr. Eng.* **2020**, *81*, 106526. [CrossRef]
8. Nakamoto, S. Bitcoin: A Peer-To-Peer Electronic Cash System. 2008. Available online: http://bitcoin.org/bitcoin.pdf (accessed on 23 January 2023).
9. Weerapanpisit, P.; Trilles, S.; Huerta, J.; Painho, M. A Decentralized Location-Based Reputation Management System in the IoT Using Blockchain. *IEEE Internet Things J.* **2022**, *9*, 15100–15115. [CrossRef]
10. AVISPA. Automated Validation of Internet Security Protocols and Applications. Available online: http://www.avispa-project.org/ (accessed on 23 January 2023).
11. MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library. Available online: https://github.com/miracl/MIRACLAccessed (accessed on 23 January 2023).
12. Ling, C.; Newport, C. Provably secure ciphertext policy ABE. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; pp. 456–465.
13. Lewko, A.; Waters, B. Decentralizing attribute-based encryption. In Proceedings of the Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011; pp. 568–588.
14. Yeh, L.Y.; Chiang, P.Y.; Tsai, Y.L.; Huang, J.L. Cloud-based fine-grained health information access control framework for lightweight IoT devices with dynamic auditing and attribute revocation. *IEEE Trans. Cloud Comput.* **2015**, *6*, 532–544. [CrossRef]
15. Miao, Y.; Ma, J.; Liu, X.; Li, X.; Liu, Z.; Li, H. Practical attribute-based multi-keyword search scheme in mobile crowdsourcing. *IEEE Internet Things J.* **2017**, *5*, 3008–3018. [CrossRef]
16. Liu, Y.; Zhang, Y.; Ling, J.; Liu, Z. Secure and fine-grained access control on e-healthcare records in mobile cloud computing. *Future Gener. Comput. Syst.* **2018**, *78*, 1020–1026. [CrossRef]
17. Ding, S.; Li, C.; Li, H. A novel efficient pairing-free CP-ABE based on elliptic curve cryptography for IoT. *IEEE Access* **2018**, *6*, 27336–27345. [CrossRef]
18. Lu, X.; Pan, Z.; Xian, H. An efficient and secure data sharing scheme for mobile devices in cloud computing. *J. Cloud Comput.* **2020**, *91*, 1–13. [CrossRef]
19. Zhang, Y.; He, D.; Choo, K.K.R. BaDS: Blockchain-based architecture for data sharing with ABS and CP-ABE in IoT. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1–9. [CrossRef]
20. Ding, S.; Cao, J.; Li, C.; Fan, K.; Li, H. A novel attribute-based access control scheme using blockchain for IoT. *IEEE Access* **2019**, *7*, 38431–38441. [CrossRef]
21. Guo, R.; Shi, H.; Zheng, D.; Jing, C.; Zhuang, C.; Wang, Z. Flexible and efficient blockchain-based ABE scheme with multi-authority for medical on demand in telemedicine system. *IEEE Access* **2019**, *7*, 88012–88025. [CrossRef]
22. Yang, X.; Li, T.; Pei, X.; Wen, L.; Wang, C. Medical data sharing scheme based on attribute cryptosystem and blockchain technology. *IEEE Access* **2020**, *8*, 45468–45476. [CrossRef]
23. Wei, X.; Yan, Y.; Guo, S.; Qiu, X.; Qi, F. Secure Data Sharing: Blockchain enabled Data Access Control Framework for IoT. *IEEE Internet Things J.* **2021**, *9*, 8143–8153. [CrossRef]
24. Qin, X.; Huang, Y.; Yang, Z.; Li, X. LBAC: A lightweight blockchain-based access control scheme for the internet of things. *Inf. Sci.* **2021**, *554*, 222–235. [CrossRef]
25. Son, S.; Lee, J.; Kim, M.; Yu, S.; Das, A.K.; Park, Y. Design of secure authentication protocol for cloud-assisted telecare medical information system using blockchain. *IEEE Access* **2020**, *8*, 192177–192191. [CrossRef]
26. Castro, M.; Liskov, B. Practical Byzantine fault tolerance and proactive recovery. *Acm Trans. Comput. Syst. (TOCS)* **2002**, *204*, 398–461. [CrossRef]
27. Macdonald, M.; Liu-Thorrold, L.; Julien, R. The blockchain: A comparison of platforms and their uses beyond bitcoin. *Work. Pap.* **2017**, 1–18. [CrossRef]
28. Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 89–98.
29. Dolev, D.; Yao, A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
30. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Advances in Cryptology–CRYPTO (Lecture Notes in Computer Science)*; Springer: Santa Barbara, CA, USA, 1999; Volume 1666, pp. 388–397.
31. Messerges, T.S.; Dabbish, E.A.; Sloan, R.H. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.* **2002**, *51*, 541–552. [CrossRef]
32. Canetti, R.; Krawczyk, H. Universally Composable Notions of Key Exchange and Secure Channels. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02), Amsterdam, The Netherlands, 28 April–2 May 2002; pp. 337–351.