

## RESEARCH ARTICLE

# Efficient, Traceable and Privacy-Aware Data Access Control in Distributed Cloud-Based IoD Systems

ZHUO MA<sup>1</sup> AND JIAWEI ZHANG<sup>1,2</sup><sup>1</sup>School of Communication Engineering, Xidian University, Xi'an, Shaanxi 710071, China<sup>2</sup>School of Cyber Engineering, Xidian University, Xi'an, Shaanxi 710126, China

Corresponding author: Jiawei Zhang (zjw8512@126.com)


This work was supported in part by the Key Research and Development Program of Shaanxi Province under Grant 2021GY-040, and in part by the Fundamental Research Funds for the Central Universities under Grant XJSJ23183.

**ABSTRACT** The emerging combination of Internet of Things (IoT) and aerospace integration aided by satellite and 6G communication techniques has stimulated the Internet of Unmanned Aerial Vehicles (UAVs), i.e., Internet of Drones (IoD). To accommodate and share the enormous real-time UAV data, cloud-based IoD is an inevitable choice to lower the heavy burden of mobile UAVs. Nevertheless, how to protect highly sensitive UAV data in such a honest-but-curious, open and distributed environment with resource-limited UAVs is a significant challenge. Although our previous work (PATLDAC) in SPNCE'21 devises a cloud-based UAV data access control scheme with policy privacy protection, limited access time and user traceability, it incurs inflexible and centralized cloud data storage and access as well as untrustworthy metadata in untrusted cloud environment for data access and user tracing. To this end, we further propose a blockchain-based privacy-aware data access control (BPADAC) scheme for distributed and secure UAV data sharing in cloud-based IoD. Based on fine-grained, traceable and privacy-preserving UAV data access characteristic of our previous work, we extend it by leveraging blockchain and Distributed Hash Table (DHT) for distributed and trustful UAV data access and storage, together with reliable and limited access mechanism to guarantee cloud UAV data sharing service provision. We also design public and undeniable user tracing mechanism to prevent user key abuse with traitor denial. Finally, we present formal security analysis and prototype the system leveraging the smart contracts of Ethereum blockchain for performance evaluation to show the feasibility of BPADAC.

**INDEX TERMS** Cloud-based IoD, blockchain, CP-ABE, hidden access policy, limited access times.

## I. INTRODUCTION

The fast growth of Internet of Things (IoT) [1] and aerospace integration with satellite and 6G communication [2] techniques recently have promoted the promising Unmanned Aerial Vehicles (UAVs) applications. The massive ubiquitous access provided by 6G ground stations (GS) [3] and powerful connection capacity among smart devices of IoT [4] facilitate the emerging Internet of Drone (IoD) [5] which enables interconnected UAVs to be deployed in various fields for task execution involving traffic supervision, disastrous rescue,

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu .

good delivery and so on. Especially, with the help of the integrated networks of satellite communications [6], [7] and ground communications, UAV groups are qualified for their tasks in more complex environment. During the IoD task completion process, collecting and tackling enormous UAV data for analysis and prediction is a heavy burden for drones with limited resources [8]. Thus, cloud-based IoD systems are dedicated to provide an ideal platform for UAV data sharing and outsourcing as it manages sufficient resources. However, the UAV data collected by drones usually has large scale and contains huge amounts of sensitive information including location-related data and GPS data [9]. Catastrophic consequence may occur if these data are

compromised in honest-but-curious cloud. Hence, security concerns of outsourced UAV data in mobile cloud-based IoD is a severe and tough challenge.

An effective way to deal with security problem of UAV data sharing in cloud-based IoD is data access control with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [10], [11], [12], [13], [14], [15], [16], [17], [18]. The approach can guarantee data confidentiality and fine-grained access control by allowing data owners to formalize specific access policies in order to indicate the privilege of data users on encrypted outsourced data in cloud. However, many severe challenges still remains in conventional CP-ABE schemes when deployed in mobile cloud-based IoD systems. Firstly, the plaintext access policies in the ciphertexts of conventional CP-ABE schemes are vulnerable to privacy leakage [19]. For instance, suppose an access policy “(SSN:10010 AND Role: caption) OR (Department: Marine Corps AND State: Philadelphia)” is set for the ciphertext in cloud-based IoD. Any one obtaining the policy can reason out the information about the users of the shared UAV data. It will be horrible for UAV application especially in military field. To this end, Zeng et al. [20] and Li et al. [21] proposed two typical schemes in standard model to effectively preserve the privacy in access policy with partially hidden access policy, but the low efficiency in UAV data encryption and decryption is intolerable.

Secondly, as UAV data from cloud-based IoD system contains large amounts of sensitive information, it may be profitable for an insider to leak these valuable data to outsider by sharing their keys, which is called key abuse attack from a traitor and leads to UAV data leakage, e.g., military secret divulgence. The problem is intractable for cloud data access control with traditional CP-ABE schemes which cannot uncover precisely a malicious insider using only his/her shared decryption key that is associated with just a set of attributes. For this problem, many researches have devised traceable CP-ABE schemes [22], [23], [24] by combining traceable mechanism with CP-ABE schemes. A typical way is white-box user tracing that integrates user identity into user decryption key such that it is easy to disclose a traitor. Whereas, many existing white-box traceable CP-ABE schemes [25], [26], [27] either cost too much computation to trace a traitor or incur heavy burden to centralized user tracing authority that maintains a list of users for private user tracing. Also, these methods cannot avoid the risk of being denied by traitor after user tracing. Thus, how to improve the efficiency of user tracing as well as uncover the traitor publicly without their denial is urgently to be solved for traceable CP-ABE when utilized in cloud-based IoD systems.

In addition, cloud-based IoD systems are located in an open environment which faces various attacks from outside, such as replay attack, impersonating attack, sniffing and intercepting attack, tampering attack and DoS (Denial of Service) attack, etc [28]. In these common attacks, DoS attack is a most fatal one that can disable data and service provision from cloud to UAV data consumers. Actually,

a malicious insider may continuously access the data sharing system to exhaust the resources of cloud and interrupt data availability such that requests of UAV data consumers will be rejected and disastrous consequence may occur especially in military field and rescue operations. Thus, this significant factor should be taken into consideration for data access control in UAV data sharing of cloud-based IoD systems. Recently, several existing CP-ABE schemes [29], [30], [31] have been raised to constrain data access frequency while incur heavy computation costs for access verification and are unsuitable for cloud-based IoD systems with resource-limited UAV devices. Moreover, UAV groups of IoD systems are usually in mobile environment and in different space from UAV data consumers, which requires distributed data storage and access. Therefore, how to deploy IoD systems in a decentralized environment with distributed, limited and fine-grained UAV data access in front of large data scale is significant in UAV data sharing of cloud-based IoD systems.

## A. CONTRIBUTIONS

Analyzing by synthesis the aforementioned problems, all of them bring great challenges to UAV data sharing service in cloud-based IoD systems. Although our previous work [32] put forward a cloud-based UAV data access control scheme (i.e., PATLDAC) which supports data confidentiality and fine-grained access control with policy privacy protection, limited access time and user traceability to solve the aforementioned problems of privacy leakage, DoS attack and user key abuse by traitors at the same time to some extent. However, it cannot support flexibly distributed data storage, the scalability of which is specifically desired in mobile IoD systems with increasingly massive UAV data. Moreover, the metadata used for data access and access time limitation faces significant security threat in cloud computing environment, which may incur privilege abuse in data access, especially the case of distributed UAV data storage. Furthermore, the traitors uncovered by PATLDAC have the opportunity to deny their malicious behaviors. Taking these issues into consideration, in this paper, we further propose a blockchain-based privacy-aware data access control (BPADAC) scheme for distributed and secure UAV data sharing in cloud-based IoD. Based on fine-grained, traceable and privacy-preserving UAV data access characteristic of our previous work PATLDAC, the superior scheme BPADAC moves forward a step to protect distributed UAV data storage and sharing in mobile cloud-based IoD by combining blockchain and Distributed Hash Table (DHT) [33] techniques for the purpose of distributed data access and storage, secure data sharing service provision, attribute privacy protection in access policy, undeniable and public traitor tracing and lower cost in computation.

Specifically, the contributions of this paper compared with our previous work is listed as follows:

- **Scalable and Distributed data storage.** To accommodate large scale and ever-increasing UAV data, traditional centralized cloud computing in most of

existing schemes and our previous work no longer works. Thus, BPADAC adopts distributed data storage containing scalable and multiple clouds. To protect its security and trustworthiness, blockchain and DHT techniques are integrated such that the chained multiple clouds can provide scalable and trustworthy resources for UAV data outsourcing. Besides, BPADAC also achieves access policy privacy protection with partially policy hiding technique (see details in Section V) as our previous work PATLDAC.

- **Distributed, limited and trustworthy data access.** Under the circumstance of distributed IoD system, UAV data that are outsourced in decentralized multiple clouds tends to be accessed in a distributed way with the help of blockchain for access control and trustworthiness guarantee. In order to ensure UAV data sharing service provision which is vulnerable to DoS attacks caused by unconstrained access aiming to deplete cloud resources, BPADAC can achieve trustable data access time limitation for each valid user by integrating blockchain and access restriction techniques, which is lacked in our previous work.
- **Undeniable and public traitor tracing, efficiency and security.** For the purpose of dealing with key abuse problem, BPADAC inherits the public white-box tracing mechanism that traitors can be uncovered by any entities of the system publicly and efficiently without the need of user list maintenance in centralized CA. However, to prevent traitors from denying the uncovered malicious behavior proof, BPADAC leverage blockchain to record immutable proofs of traitors for tracing consistency. Moreover, with thorough efficiency analysis by large number of experiments, BPADAC shows its higher performance both in data encryption and decryption as a result of online/offline encryption and outsourced decryption testing techniques. We also present formal security model with corresponding proofs for BPADAC which is not given in our previous work.

## B. OUTLINE

The rest of this paper is organized as follows. In Section II, we give some background of related work. Section III introduces preliminaries for the proposed scheme. The system model and threat model as well as the formal definition of the proposed scheme are given in Section IV. The detailed workflow and construction together with security analysis of our scheme is shown in Section V. We provide thorough performance analysis in Section VI. Section VII gives the conclusion of our work.

## II. RELATED WORK

It has been widely studied that data assets play an important role in UAV applications for analysis and prediction [40], [41], [42], [43], [44], [45]. As one type of highly valuable assets, UAV data security is more and more severe. To this end, Tsao et al. [8] proposes a secure UAV transmission

system for UAV-to-UAV communication protection. Moreover, Alladi [46] designs a UAV-to-UAV authentication scheme to identify end-to-end communication between UAVs and ground stations. Besides, Mehta et al. [47] intends to protect UAV networks security by combining 5G-enabled UAV system with blockchain. However, these proposals are helpless in front of the challenging data access control problem for UAV data sharing in cloud-based IoD systems.

In the research field of data access control, CP-ABE [10], [11], [12], [48], [49], [50] is widely accepted in various data sharing scenario in honest-but-curious public data storage, such as cloud data sharing, to achieve data confidentiality and fine-grained access control. Although CP-ABE is a effective and helpful tool in data security, its heavy overhead in computation of encryption and decryption is still undesirable in many applications. To this end, Hohenberger and Waters [36] first integrated online/offline computing approach into CP-ABE and proposed an online/offline CP-ABE scheme. Xue et al. [51] utilized this approach in multi-authority CP-ABE and designed an OOMA-CP-ABE scheme, which also gives birth to the scheme proposed in [35] combining online/offline into CP-ABE. From the view of computation cost mitigation in decryption, the idea of outsourced decryption was first designed by Green et al. [52] which is introduced in many recent works in CP-ABE. Borrowing this efficient paradigm and aiming to guarantee the correctness of the results in outsourced decryption, Lai et al. [34] proposed a verifiable outsourced CP-ABE scheme to verify the outsourced decryption results returned by malicious cloud servers. Further, it is natural that many recently devised CP-ABE schemes utilize both online/offline encryption and outsourced decryption approaches in CP-ABE for higher efficiency, such as the proposal in [53].

Moreover, traditional CP-ABE schemes are unable to avoid privacy leakage in access policy associated with shared ciphertexts and thus cannot be used in sensitive data sharing applications, such as UAV data sharing and healthcare data sharing. To deal with the problem, a partial hidden policy CP-ABE scheme [54] was proposed to guarantee policy security with fully secure proofs. For expressiveness, Lai et al. [39] constructed another CP-ABE scheme with expressive and partial hidden access policy, meantime Zhang et al. [19] devised a privacy-preserving CP-ABE scheme with both expressive and hidden policy over large attribute universe which also achieves full security under standard model. Nevertheless, the above schemes cannot solve user key abuse problem that is hazardous in data leakage by third party. To deal with this challenge, Li et al. [21] proposed a traceable and privacy-preserving CP-ABE scheme with full security by borrowing the white-box user tracing mechanism introduced in [19]. However, the scheme relies heavily on a centralized CA and user list for user tracing. For mitigating this reliance, Zeng et al. [20] combined the public user tracing approach used in [37] and [38] to design a privacy-preserving and publicly traceable CP-ABE scheme. Whereas, its efficiency

TABLE 1. Function comparison in various schemes.

Scheme	DS	PH	LU	TLDAC	FS	SM	OOE	ODT	VR	PT
Scheme [34]	×	✓	×	×	✓	✓	×	×	×	×
Scheme [35]	×	×	×	×	✓	×	✓	×	×	×
Scheme [36]	×	×	×	×	×	×	✓	×	×	×
Scheme [30]	×	×	✓	✓	×	×	×	×	×	×
Scheme [37]	×	×	✓	×	×	×	×	×	×	✓
Scheme [38]	×	×	✓	×	×	×	×	×	×	✓
Scheme [39]	×	✓	×	×	✓	✓	×	×	×	×
Scheme [19]	×	✓	✓	×	✓	✓	×	×	×	×
Scheme [21]	×	✓	✓	×	×	×	✓	×	×	×
Scheme [20]	×	✓	✓	×	✓	✓	×	×	×	✓
PATLDAC	×	✓	✓	×	✓	✓	✓	×	✓	×
BPADAC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note. DS: distributed storage; PH: policy hiding; LU: large universe; TLDAC: trustworthy time-limited data access; FS: full security; SM: standard model; OOE: online/offline encryption; ODT: outsourced decryption test; VR: verifiability; UDPT: undeniable and public traceability.

in computation is too high to be suitable for resource-limited devices.

Furthermore, in the era of IoD, drones generate large scale real-time UAV data to be analyzed and utilized. Storing this type of massive data is a big challenge. The resources of single cloud in cloud-based IoD is short of scalability and may be depleted by continuously generated UAV data. Thus, distributed multi-cloud storage is a must for stronger resource elasticity and scalability. Li et al. [33] made contributions to this problem by introducing DHT and blockchain technique into distributed storage, while Ren et al. [55] devised a multi-cloud storage mechanism for smart homes based on blockchain. Meanwhile, distributed data access control in many types of cloud-based IoT systems becomes a complex challenge. Although Roy et al. [56] proposed a fine-grained data access control scheme in multiple clouds environment, they failed to solve the problem of distributed data access. Therefore, Li et al. [57], Feng et al. [58] and Liang et al. [59] put forward several blockchain-based approaches to deal with distributed data access in multiple clouds. Based on this idea, blockchain is widely adopted in distributed and secure data access control scenario and inspires several following fine-grained and distributed data access control schemes [60], [61], [62], [63], [64]. However, these proposals scarcely take multiple and distributed cloud storage into consideration.

To address the above challenges, based on our previous work PATLDAC [32], we present a blockchain-based privacy-aware data access control (BPADAC) scheme for distributed and secure UAV data sharing in cloud-based IoD setting based on the blockchain and DHT-based distributed storage and data access as well as public and undeniable traceability together with trustworthy access time limitation and attribute privacy protection technique. We make a comparative summary of our scheme BPADAC, our previous scheme PATLDAC and some existing schemes in TABLE 1.

### III. PRELIMINARIES

#### A. NOTATIONS

Throughout the paper, we use  $[l1, l2]$  as the set  $\{l1, l1 + 1, \dots, l2\}$  and  $[n]$  as the set  $1, 2, \dots, n$ , where  $n \in \mathbb{Z}_p^*$ .

TABLE 2. Notation descriptions in BPADAC.

Notations	Descriptions
$PK, MSK$	system public parameter and master key
$PK_u, SK_u$	user public key and secret key
$ID_u, S$	user identity and attribute set
$I_s, S$	index and value of each attribute in set $S$
$DK_u, TK_u$	user decryption key and transformation key
$IT_t, CT$	intermediate ciphertext and final ciphertext
$P_0, P_1$	components of partial decrypted ciphertext
$T_X$	the transaction for action "X" of blockchain

Besides,  $|S|$  is used to denote the length of a string  $S$  and some of the other key notations are given in Table 2.

#### B. BLOCKCHAIN AND SMART CONTRACT

Blockchain (i.e., BC) [59] is formed as a decentralized and distributed immutable ledger based on peer-to-peer network infrastructure. Each node of this peer-to-peer network stores a copy of the ledger of blockchain, while the consistency of these distributed ledgers is built on certain consensus mechanism including proof of work (PoW) implemented in Bitcoin such that the untrusted parties can formulate decentralized trustworthiness among themselves. The ledger contains a chain of chronologically linked data blocks each of which is composed of a block header and a series of transactions.

Beyond cryptocurrencies, the most commonly used application of blockchain in diverse fields is smart contract. Smart contract is a specific protocol containing a series of logic computations executed on the blockchain under predefined conditions in essence. It is deployed in blockchain and its results can be self-executed and self-verified without human intervention. Thus, smart contract is actually a kind of computer program and makes blockchain programmable. The results of a smart contract are immutable and credible.

Generally, blockchain is classified into permissionless and permissioned chains, of which the former allows pseudonymous or even anonymous participants. Each block of this ledger are visible to all entities in the network which drives

all nodes to participate in consensus. The computations are performed by each node so that the computation resources of each node in blockchain become the bottleneck. For instance, the throughput of a permissionless blockchain is on the order of hundreds of transactions per second. A permissioned blockchain is built with a certain level of trust among users.

Our scheme is built on Ethereum which is an open source blockchain platform derived from Bitcoin. It supports pluggable consensus algorithm including POW as well as Turing complete and flexible smart contracts. Ethereum has two kind of accounts, that is, External Owned Accounts (EOA) and Contract Accounts. The former is owned by external users of blockchain with their private keys. It contains balance and Nonce field, and can send a transaction to another address or trigger the execution of the contract code. The latter is owned by the smart contract code deployed in blockchain. Besides the balance and Nonce fields, contract account has associated storage space and corresponding code executed on Ethereum Virtual Machine (EVM) which is a smart contract running environment. Similarly to Bitcoin, ether is the token used in the Ethereum platform. The consumption for the operation in EVM is counted by the unit gas purchased via ether. The issuer of a transaction has to pay for the operation he desires (e.g., data storage, computing) with ether. Actually, the cost of a transaction is computed as ether = Gas Used × Gas Price. Particularly, in our BPADAC, we stored the state of multiple cloud and part of ciphertext metadata in blockchain deploying two smart contracts used for outsourced decryption testing and undeniable public user tracing.

C. ACCESS STRUCTURE

Definition 1 (Access Structures [11]): Suppose there is a collection of entities  $M = \{M_1, \dots, M_n\}$  and a set  $E \subseteq 2^M \setminus \emptyset$ , we say  $E$  is monotonic in case that  $\forall F, G : F \subseteq G \cap F \in E \rightarrow G \in E$ . We also denote the set  $E$  as a monotonic access structure and its subsets as the authorized sets, otherwise, the unauthorized sets.

D. LINEAR SECRET SHARING SCHEMES (LSSS)

Definition 2 (LSSS [19]): We use  $P$  to denote the attribute universe of CP-ABE scheme. Each attribute in  $P$  has two properties, that is, attribute name and attribute value that may be multivalued. Let  $\mathcal{M}$  be an  $l \times n$  matrix over  $Z_p$ , an LSSS over  $P$  can be denoted by  $(\mathcal{M}, \rho)$  on  $P$  and  $\mathcal{M}$  is named share-generating matrix while  $\rho$  is a map from each row of  $\mathcal{M}$  to the index of each attribute name. Suppose there is a vector  $v = (s, y_2, \dots, y_n)^T$ , where  $s \in Z_p$  is the secret to be shared and  $y_2, \dots, y_n \in_R Z_p$ , then  $\lambda_x = \mathcal{M}_x \cdot v$  is a component of the shared secret  $s$  in correspondence to the attribute indicated by  $\rho(x)$ . Assuming  $I = \{i | \rho(i) \in E\} \subseteq \{1, 2, \dots, l\}$ , where  $E$  is an authorized set. We can compute a set of coefficients  $\{w_i \in Z_p\}_{i \in I}$  to satisfy  $\sum_{i \in I} w_i \mathcal{M}_i = (1, 0, \dots, 0)$ . Thus,  $I$  is regarded as a minimum authorized set of  $(\mathcal{M}, \rho)$  in case that it meets  $(\mathcal{M}, \rho)$  while any  $I' \subset I$  does not meet  $(\mathcal{M}, \rho)$ .

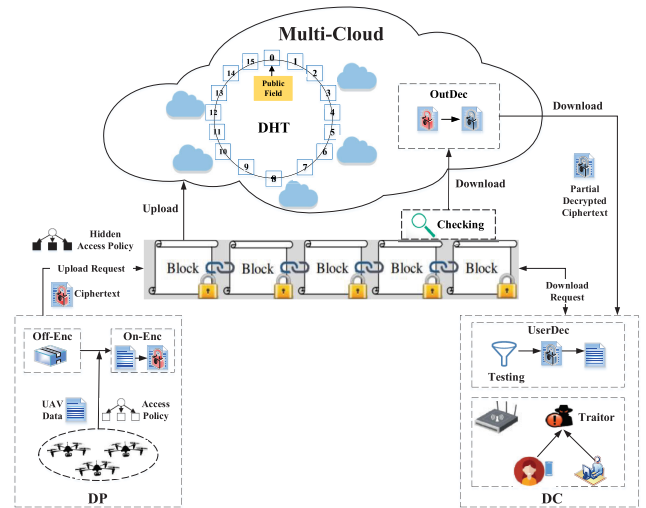


FIGURE 1. The system model of our BPADAC.

E. COMPOSITE BILINEAR MAP

Definition 3 (Composite Bilinear Maps [10]): Suppose a group generator  $\mathcal{B}$  taking a security parameter  $\lambda$  as inputs and outputing a bilinear group  $G = (N, p_1, p_2, p_3, p_4, G, G_T, \hat{e})$ , where  $G = G_{p_1} \times G_{p_2} \times G_{p_3} \times G_{p_4}$  and  $p_1, p_2, p_3, p_4$  are different prime numbers while  $G, G_T$  are cyclic groups of composite order  $N = p_1 p_2 p_3 p_4$  and  $\hat{e} : G \times G \rightarrow G_T$  is a composite bilinear map satisfy the following properties:

- Bilinearity:  $\hat{e}(f^c, q^d) = \hat{e}(f, q)^{cd}, \forall c, d \in Z_N, f, q \in G$ .
- Non-Degenerate:  $\exists f \in G \rightarrow \hat{e}(f, f) \in G_T \cap \text{ord}(\hat{e}(f, f)) = N$ .
- Computability: with respect to  $\forall f, q \in G, \hat{e}(f, q)$  is efficient in computation.

Assuming  $G_{p_i}$  is the subgroup of prime order  $p_i$ , then  $\forall i \neq j \cap X_i \in G_{p_i} \cap X_j \in G_{p_j} \rightarrow \hat{e}(X_i, X_j) = 1$ . We say these subgroups are “orthogonal” to each other.

IV. SYSTEM MODEL AND FORMAL DEFINITION

In this section, we describes the system model of BPADAC together with its threat model involving possible attacks and security requirements as well as the formal definition.

A. SYSTEM MODEL

Fig.1 shows the system model of our proposed BPADAC. The system involves five generic entities, that is, Trusted Authority (TA), UAV Cloud Providers (UCPs), Blockchain (BC), Data Producer (DP) and Data Consumer (DC). Their functionalities are described in detail below.

- TA is responsible for initializing the system and user registration and authorization. When register users of the system, TA distributes their decryption keys and transformation keys after user identity authentication. Being successfully registered in TA, each user can join in the blockchain network.
- UCP is the multi-cloud that is dedicated in providing users with various and heterogenous UAV data services

containing unlimited computing and storage resources. It includes multiple clouds each of which belongs to different service provider and is randomly allocated a unique address by DHT. When users uploading or downloading files, the address is used to locate the specific cloud service provider in UCP.

- BC is the blockchain that provides immutable data storage, fair and distributed access framework and so on. In our scheme, BC is used to store public parameters and limit data access time to prevent UCP services from being attacked by DDoS, etc. Besides, BC is a transfer station between users and UCP in data outsourcing and access. BC can also be used in public user tracing, which can record unchangeable proof of malicious user intending to leak their keys for illegal profit.
- DP includes various UAVs that can generate or collect massive spatiotemporal UAV data. To save storage cost for resource limited UAVs, the data should be outsourced to UCP via BC which will record the uploading proofs of the outsourced data including their identities and addresses of cloud service provider accommodating these data.
- DC represents UAV data consumers that wants to analyze and mines deep information with machine learning related approaches. Privileged DCs can have rights to distributedly access the desired UAV data shared in UCP through BC. Also, BC will record data downloading proofs. What is more, BC will check if the access times of the UAV data exceeds its maximum and records the keys of malicious DCs for undeniable and public user tracing.

## B. THREAT MODEL

In our system, TA is deemed as a fully trusted entity since it is in charge of generating system parameters and issuing keys for users through secure channel. UCP is considered to be semi-honest that carries out data outsourcing and access actions faithfully while may intend to perform some passive attacks. DP is assumed to be fully trusted who will generate massive UAV data and outsource it to UCP. It has no motivation to launch attacks to the system. DC is untrusted part of the system because the UAV network organization is open complex environment. Any entity may take part in the system to share the UAV data without any privilege. Even an authorized DC (i.e., insider DC) may maliciously leak their keys for illegal profit and deny any evidence. Thus, the following part summarizes the possible attacks towards our system.

### Possible attacks:

- *Eavesdropping attack*: Any entity may eavesdrop the data transferred in wireless UAV network to learn critical information even access shared data in UCP without any authorization or collusion with several other unauthorized entities.
- *Data tampering attack*: The attack means that the providers in UCP may tamper the decryption results

when DC access UAV data shared in UCP which may affect the local decryption process of DC.

- *Privacy leakage attack*: Any entity that can approach the shared UAV data in UCP may infer more sensitive information from the access policy associated with ciphertext in UCP, which may cause significant data leakage.
- *Key abuse attack*: The attack can be launched by any malicious insider, that is, any authorized DC can share their keys to outsiders for illegal profit.
- *DDoS attack*: In any open complex environment, especially Internet-based cloud network, the data service provider is vulnerable to DDoS attack. Any entity can join in the network of the multiple clouds in UCP to launch the attack towards specific cloud service provider and disable the data services.

### Security Requirements:

- *Fine-grained data access control and confidentiality*: Aiming at the first possible attacks, our system should ensure data confidentiality primarily to protect sensitive information in UAV data and prevent unauthorized data access.
- *Data integrity protection*: In our system, we should verify if the result of outsourced decryption returned from UCP is correct or compatible with DC according to the second possible attack.
- *Privacy protection in access policy*: With respect to the third possible attack, the data information of the attributes in access policy should be taken into consideration in our system.
- *Public user tracing for malicious insiders*: In view of the fourth possible attack, any entity in our system should be able to find out the real identity of malicious insiders once abnormal actions is detected. In the meantime, the identified traitor cannot deny the solid evidence.
- *Constraint data access times for authorized DCs*: Considering the last possible attack, we should make sure that our system can check the data access time before UCP providing outsourced decryption.

## C. FORMAL DEFINITION OF BPADAC

The following procedures describes the definition of the proposed BPADAC, which is a privacy-preserving and public tracing CP-ABE scheme supporting online/offline encryption and outsourced decryption with outsourced testing as well as data access time checking.

- $Setup_S(\lambda) \rightarrow (PK, MSK)$ : The procedure is execute by TA. On inputting the security parameter  $\lambda$ , the procedure publishes system public parameter  $PK$  and master key  $MSK$ . Besides, TA initiates the blockchain network of BC and deploys user tracing smart contract in BC.
- $Setup_C(PK) \rightarrow (ctr, L, ST)$ : The procedure is executed by UCP. It initiates the variance and state lists used for data access time limitation. Then, UCP takes part in BC and stores the state list into BC through a transaction.

- $Setup_U(PK) \rightarrow (PK_u, SK_u)$ : The procedure is run by DP/DC. It takes system public parameter  $PK$  as inputs and outputs public key  $PK_u$  and secret key  $SK_u$  for each entity including DC and DP. Then, DP/DC takes part in BC.
- $KeyGen(PK, MSK, PK_u, ID_u, S) \rightarrow DK_u$ : The procedure is run by TA. On inputting the system public parameter  $PK$  and master key  $MSK$ , the public key  $PK_u$  of the data user and his attribute set  $S$  with identity  $ID_u$ , the procedure outputs the decryption key  $DK_u$ .
- $KeyGen_{OUT}(PK, DK_u, SK_u, csi) \rightarrow TK_u$ : Given the system public parameter  $PK$ , the decryption key  $DK_u$  and the user secret key  $SK_u$  of a data user and the current state information  $csi$  which is a string describing the state of system, the algorithm outputs transformation key  $TK_u$  for the data user.
- $Encrypt_{off}(PK) \rightarrow IT_t$ : The procedure is run by DP. Given the system public parameter  $PK$ , the procedure outputs the intermediate ciphertext  $IT_t$ .
- $Encrypt_{on}(PK, IT_t, M, A) \rightarrow CT$ : The procedure is executed by DP. According to the system public key  $PK$  and access policy  $A$ , the procedure computes the ciphertext of UAV data  $M$  identified by  $ID_m$  assisted by intermediate ciphertext  $IT_t$ . As the owner of ciphertext, DP deploys a data access time checking smart contract in BC used in data access to check if data access time exceeds the maximum value and execute outsourced decryption testing. Then, the ciphertext  $CT$  is uploaded to UCP after a data uploading transaction is verified consistently by BC.
- $Decrypt_O(PK, CT, TK_u) \rightarrow (P_0, P_1, CT)$ : The procedure is executed by UCP and BC together. After receiving the data access request from DC through a transaction to BC, the writing node of BC triggers a smart contract to check the data access time limitation and decryption testing. Then, UCP initiates ciphertext outsourced decryption with system public parameter  $PK$ , ciphertext  $CT$  and transformation key  $TK_u$ , and outputs the partially decrypted ciphertext  $(P_0, P_1, CT)$ .
- $Decrypt_U(PK, SK_u, (P_0, P_1, CT)) \rightarrow M$ : The procedure is executed by DC. After receiving the partially decrypted ciphertext  $(P_0, P_1, CT)$  with system public parameters  $PK$ , the DC decrypts with user secret key  $SK_u$  and outputs the plaintext.
- $UserTrace(PK, DK_u) \rightarrow ID$  or  $\perp$ : The procedure is executed by any entity of the system. After catching the leaked decryption key  $DK_u$  of malicious DC, any entity can launch a user tracing transaction to BC to trace malicious user publicly with a smart contract on inputting system public parameter  $PK$  and outputs the identity  $ID$  of the malicious DC or  $\perp$  undeniably.

V. PROPOSED SCHEME

A. WORKFLOW OF BPADAC

This part shows the workflow of our BPADAC that deploys the basic construction tool as referred in Fig.2. The system

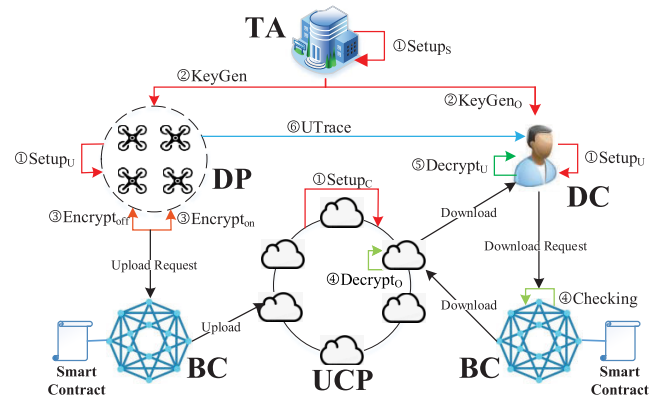


FIGURE 2. The workflow of our BPADAC.

has five phases, that is, system initialization phase, entity registration phase, data outsourcing phase, data access phase and user tracing phase.

1) SYSTEM INITIALIZATION

This phase launches the process of system setup to initialize system parameters. As shown in Fig.2, in step ①, TA, UCP and DP/DC runs  $Setup_S$ ,  $Setup_C$  and  $Setup_U$  algorithms respectively to generate system parameters, UCP parameters and user parameters, respectively. Besides, TA also initiates the blockchain network BC and deploys user tracing smart contract in this phase.

2) ENTITY REGISTRATION

The step ② in Fig.2 runs  $KeyGen$  and  $KeyGen_O$  algorithms for each entity including DP and DC to distribute their decryption keys and transformation keys, respectively. After registration, each entity successfully joins in the blockchain network BC to issue transactions verified by BC and take advantage of the immutable property of BC for necessary data or metadata storage.

3) DATA OUTSOURCING

Fig.2 describes the data outsourcing process in step ③. DP encrypts the UAV data according to designated access policy by running  $Encrypt_{off}$  and  $Encrypt_{on}$  algorithms to mitigate the computation cost of encryption and outsources it with associated policy to UCP through BC by submitting corresponding transaction to achieve distributed data outsourcing. The attribute values of access policy are hidden in encrypted UAV data while only attribute names are revealed. Besides, DP deploys a data access time checking smart contract to BC in this phase.

4) DATA ACCESS

Fig.2 depicts the process of data access. When downloading desired files shared in UCP, DC has to submit a transaction to BC for data access with an interaction between DC and BC for data access time checking and outsourced decryption testing performed by the data access time checking smart contract of BC. After successful data access request checking

by BC, UCP executes outsourced decryption algorithm  $Decrypt_O$  according to the data access request from DC and returns partial decrypted ciphertext to DC (See step ④). Then, privileged DC can get plaintext of the file by running user decryption algorithm (See step ⑤).

## 5) USER TRACING

Fig.2 describes the phase of user tracing in step ⑥. When a malicious DC executes abnormal operations, any entity of the system can submit a transaction to BC and trigger the user tracing smart contract so as to disclose the identity of the traitor publicly with the help of BC. Meanwhile, the anomalous evidence will be stored by BC without the risk of being tampered to avoid denial of the malicious DC.

In our proposed scheme BPADAC, we observe that traditional ABE-based data sharing systems fall short of protection for policy-related ciphertext components (i.e., metadata of ciphertext). The white-box tracing mechanism also incurs in proof denial from malicious users. Whereas, blockchain builds up a trustworthy system among different and untrusted entities including multiple clouds of UCP and DC. It can support distributed data access to multiple clouds in UCP as well as undeniable malicious user tracing and trustworthy outsourced decryption with decryption matching for our designed basic CP-ABE construction, which significantly advances functionality and secure property of traditional ABE-based data sharing systems facing with large scale IoD environment. When combining ABE and blockchain, an inevitable challenge is the metadata storage overhead of ABE scheme in blockchain and its efficiency in consensus. In our proposal, we greatly limit the storage overhead of metadata in blockchain and make it efficient in both storage and computation for blockchain nodes.

### B. CONCRETE CONSTRUCTION OF BPADAC

- $Setup_S(\lambda) \rightarrow (PK, MSK)$ : TA creates a bilinear group  $(N = p_1 p_2 p_3 p_4, G, G_T, \hat{e})$  by running bilinear group generator algorithm  $\mathcal{G}(\lambda)$  with security parameter  $\lambda$ , where  $\{p_i\}_{i \in [4]}$  are four primes and  $G, G_T$  are two  $N$ -order cyclic groups with a generator  $g \in G$  while  $\hat{e} : G \times G \rightarrow G_T$  is a corresponding bilinear map. TA picks  $\alpha, a \in_R Z_N, f, h \in_R G(p_1), A_3 \in_R G_{p_3}, O, A_4 \in_R G_{p_4}$  to generate  $B = \hat{e}(g, g)^\alpha, F = fO$  and a hash function  $H_m : \{0, 1\}^* \rightarrow Z_N$  meantime initializes the attribute universe as  $U = Z_N$ . TA returns the system public key as  $PK = (N, g, g^\alpha, h, B, F, A_4, H_m)$  publicly which is stored also in BC and stores the master key  $MSK = (\alpha, f)$  secretly. Finally, TA builds up an initial blockchain network of BC to formulate a distributed and trustworthy system among untrusted entities, and deploys a user tracing smart contract which is specifically described later.
- $Setup_C(PK) \rightarrow (ctr, L, ST)$ : UCP initializes its public/secret key pair as well as the counter  $ctr = 0$  to count data access times and the state set  $ST = \emptyset$  with system public key  $PK$  for data access time limitation

towards each DC. UCP then creates an empty list  $L$  for  $ctr$  and  $ST$  maintenance.

After setup, UCP together takes part in the blockchain network and is able to take advantage of the blockchain to store its state list  $L$  with a transaction signed by their secret key. Without loss of generality, a transaction in our scheme includes the identity of an entity, a timestamp and an action. For instance,  $T = (ID_c, TS, Action, Sig_t)$ , where  $ID_c$  is the identity of UCP,  $TS$  is the timestamp of the transaction creation,  $Action$  is the execution that UCP claims including state storage, and  $Sig_t$  is the signature of the transaction  $T$  signed by the secret key of UCP.

To be specific, when UCP stores its state list  $L$  to BC, it issues following transaction  $T_S$ :

$$T_S = (ID_c, TS, Action = \text{"store state list } L'', Sig_t),$$

where  $ID_c$  is the identity of UCP,  $TS$  is the timestamp and  $L$  is its state list. When  $T_S$  is verified by the nodes of BC with Algorithm 1, it is written into a new block appended with the state list  $L$ .

---

#### Algorithm 1 Transaction Verification

---

**Input:**  $PK_u$ : the public key of the transaction issuer,  $T$ : the transaction to be verified.

**Output:** *True/False*.

- 1:  $H = Ver(PK_u, T, Sig_t)$ , where  $Ver$  is the designated signature verification algorithm.
  - 2: **if**  $H = Valid$  **then**
  - 3:     **return** *True*.
  - 4: **else**
  - 5:     **return** *False*.
  - 6: **end if**
- 

- $Setup_U(PK) \rightarrow (PK_u, SK_u)$ : DC randomly picks  $z_u \in_R Z_N$  as secret key  $SK_u = z_u$ , and computes user public key  $PK_u = h^{z_u}$  with system public key  $PK$ . DP/DC will join in the blockchain network after setup and utilize the blockchain to store metadata, access cloud data or even trace malicious user with a transaction  $T = (ID_u, TS, Action, Sig_t)$  signed by their secret key. The  $Sig_t$  in a transaction issued by DP/DC is the signature of the transaction  $T$  signed by the secret key of DP/DC identified by  $ID_u$ .
- $KeyGen(PK, MSK, PK_u, ID_u, S) \rightarrow DK_u$ : Suppose a DP/DC associated with his/her attribute set  $S = (I_s, S)$ , where  $I_s \subset Z_N$  is the attribute name index and  $S = \{s_i\}_{i \in I_s}$  is the corresponding attribute value set. When distributing decryption key for DP/DC according to his/her identity  $ID_u$  and attribute set  $S = (I_s, S)$ , TA picks random number  $t \in Z_N$  and elements  $R, R', R_i \in G_{p_3}$  for  $i \in I_s$ . TA returns  $DK_u = \{S, K, K', K'', \{K_i\}_{i \in I_s}\}$  to DP/DC as his/her decryption key of DC, where

$$K = g^\alpha g^{atH_m(K', K'')} R, K' = g^t R', K'' = ID_u,$$

$$K_i = (g^{s_i f})^t R_i$$



- $KeyGen_O(PK, DK_u, SK_u, csi) \rightarrow TK_u$ : DP/DC generates his/her transformation key for outsourced decryption when access desired data by using his/her secret key  $SK_u$  together with the system public key  $PK$ , decryption key  $DK_u$  and current state information  $csi$  to compute  $TK_u^1 = \{I_s, K^1 = g^{z_u} \cdot K = g^{z_u} g^\alpha g^{atH_m(K', K')}, K', K'', \{K_i\}_{i \in I_s}\}$ . Then, the DP/DC calculates rest part  $K_c = B^{1/(z_u + H_m(csi))}$  which is the VRF output of  $csi$ ,  $K_p = g^{1/(z_u + H_m(csi))}$  which is the correctness proof of  $csi$ . DP/DC finally returns  $TK_u = (TK_u^1, K_c, K_p, csi)$  as his transformation key.
- $Encrypt_{off}(PK) \rightarrow IT_i$ : To mitigate the computation cost in encryption, DP computes necessary ciphertext components in advance. After selecting random values  $s, s' \in Z_N$  as shared secret value with system public key  $PK$ , DP calculates  $\tilde{C}'_\delta = B^{s'}$ ,  $\tilde{C}'_1 = B^s$ ,  $\tilde{C}'_\delta = g^{s'}$ ,  $\tilde{C}'_1 = g^s$  to generate intermediary pool  $IT_1 = \{(s, s', \tilde{C}'_\delta, \tilde{C}'_1, \tilde{C}'_\delta, \tilde{C}'_1)\}$ . Then, by choosing  $\lambda', t', r' \in_R Z_N$  and calculating  $C'_{\delta,x} = g^{a\lambda'}$ ,  $C'_{1,x} = g^{a\lambda'} (g^{t'} F)^{r'}$ ,  $C'_{2,x} = g^{r'}$ , DP generates another intermediary pool  $IT_2 = \{(\lambda', t', r', C'_{\delta,x}, C'_{1,x}, C'_{2,x})\}$ . DP returns an intermediate ciphertext  $IT_i = \{IT_1, IT_2\}$  which is used for following online encryption.
- $Encrypt_{on}(PK, IT_i, M, A) \rightarrow CT$ : When DP generates large amounts of UAV data, to save local storage, DP has to upload these data to UCP. Considering data security, DP encrypts the UAV data  $M$  and appoints desired privileges by means of access policy  $\mathcal{A} = \{A, \rho, T\}$ , where  $A$  is a  $l \times n$  share-generating matrix and  $T = \{t_{\rho(1)}, \dots, t_{\rho(l)}\}$  is the value set of the access policy  $\mathcal{A}$ . Assisted by the intermediate ciphertext  $IT_i$ , DP generates the ciphertexts of  $M$  as below.
  - DP randomly picks  $(s, s', \tilde{C}'_\delta, \tilde{C}'_1, \tilde{C}'_\delta, \tilde{C}'_1)$  from  $IT_1$  to generate two n-dimension vectors over  $Z_N$ , i.e.,  $v = (s, v_2, \dots, v_n)$ ,  $v' = (s', v'_2, \dots, v'_n)$ , where  $s, s' \in_R Z_N^n$ .
  - DP selects  $l$  different random tuples  $\{(\lambda'_x, t'_x, r'_x, C'_{\delta,x}, C'_{1,x}, C'_{2,x})\}_{x \in [l]}$  from  $IT_2$  and picks  $O_\delta \in_R G_{p_4}$ ,  $O_{\delta,x}, O_{c,x}, O_{d,x} \in_R G_{p_4}$ , where  $1 \leq x \leq l$ .
  - DP computes the ciphertext  $CT = \{(A, \rho), \tilde{C}_\delta, \hat{C}_\delta, \{C_{\delta,x}\}_{1 \leq x \leq l}, \tilde{C}_1, \hat{C}_1, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}, C_{5,x}\}_{1 \leq x \leq l}\}$ , where

$$\begin{aligned} \tilde{C}_\delta &= \tilde{C}'_\delta, \hat{C}_\delta = \tilde{C}'_\delta \cdot O_\delta, \\ C_{\delta,x} &= C'_{\delta,x} \cdot (g^{t_{\rho(x)}} F)^{-s'} O_{c,x}, \\ \tilde{C}_1 &= M \cdot \tilde{C}'_1, \hat{C}_1 = \tilde{C}'_1, C_{1,x} = C'_{1,x} \cdot O_{c,x}, \\ C_{2,x} &= C'_{2,x} \cdot O_{d,x}, C_{3,x} = A_x \cdot v - \lambda'_x, \\ C_{4,x} &= A_x \cdot v' - \lambda'_x, C_{5,x} = r'_x (t_{\rho(x)} - t'_x) \end{aligned}$$

After obtaining the ciphertext  $CT$ , DP sends data uploading request to BC by issuing the following transaction  $T_D$ :

$$T_D = (ID_u, TS, Action = "upload ciphertext ID_m \text{ to cloud server } Addr'', Sig_t),$$

where  $ID_u$  is the identity of DP,  $ID_m$  is the identity of the outsourced UAV data  $M$ ,  $TS$  is the timestamp and  $Addr$  is the cloud server address in UCP allocated by DHT. When  $T_D$  is verified by the nodes of BC with Algorithm 1, it is written into a new block appended with the part of the ciphertext  $(\tilde{C}_\delta, \hat{C}_\delta, \{C_{\delta,x}\}_{1 \leq x \leq l})$  of  $CT$  together with its hidden policy  $(A, \rho)$ . As the owner of UAV data, DP deploys a data access time checking smart contract to BC which is used later in  $Decrypt_O$  algorithm. Subsequently, DP is allowed to outsource  $CT$  to UCP offline for data sharing.

**Note:** The access policy  $(A, \rho)$  in ciphertext  $CT$  eliminates the attribute set  $T$  of the designated access policy  $\mathcal{A} = \{A, \rho, T\}$  used by DP in encryption process as  $T$  contains large amounts of sensitive information. Thus, the scheme achieves policy privacy preserving by partially hiding the attribute set of the access policy associated with outsourced ciphertext while only the attribute name index  $\rho$  remains such that any attackers cannot obtain the attribute set  $T$  from associated access policy of ciphertext  $CT$ .

- $Decrypt_O(PK, CT, TK_u) \rightarrow (P_0, P_1, CT)$ : This algorithm takes following interactive steps.

- **Data request.** When DC desires to access designated ciphertext  $CT$ , he/she first generates a data access request of  $CT$  including the ciphertext identity  $ID_m$  of  $CT$  and user identity  $ID_u$  of the DC by issuing the data access transaction  $T_A$  as below:

$$T_A = (ID_u, TS, Action = "access to ciphertext ID_m \text{ with transformation key } TK_u \text{ from cloud server } Addr'', Sig_t),$$

where  $Addr$  is the location of the cloud server in UCP allocated by DHT that stores the ciphertext  $CT$  with identity  $ID_m$ .

- **Access time checking and decryption testing.**

After being successfully verified by the nodes of BC with Algorithm 1, the data access transaction  $T_A$  triggers the data access time checking smart contract (abbreviated as DATC-SC) in Algorithm 2 to check if the access time of ciphertext  $ID_m$  has exceeds its maximum and if the ciphertext  $CT$  matches with the user request of the DC.

If the smart contract  $DATC - SC$  returns *True*, BC searches the part of the ciphertext  $(\tilde{C}_\delta, \hat{C}_\delta, \{C_{\delta,x}\}_{1 \leq x \leq l})$  with related hidden access policy  $(A, \rho)$  and executes the following equation by finding out a subset  $\mathcal{I} \subset I_{A,\rho}$  that satisfies  $\{\rho(i) | i \in \mathcal{I}\}$ ,

$$P_0 = \hat{e} \left( \prod_{i \in \mathcal{I}} C_{\delta,i}^{w_i}, (K')^{H_m(K', K'')} \right) \cdot \hat{e}(\hat{C}_\delta, K^{-1} \prod_{i \in \mathcal{I}} K_{\rho(i)}^{w_i}),$$

where  $I_{A,\rho} \subset \{1, 2, \dots, l\}$ , while  $(A, \rho)$  is the hidden access policy of  $CT$  and  $\sum_{i \in \mathcal{I}} \omega_i A_i =$

**Algorithm 2** Pseudo-Code of *DATC – SC(PK, TK<sub>u</sub>, L, ε)*

**Input:** *PK*: system public key, *TK<sub>u</sub>*: transformation key of the DC *ID<sub>u</sub>*, *L*: state maintenance list, *ε*: maximum UAV data access time.

**Output:** True/False.

- 1: Fetch the tuple (*ctr*, *ST*) from *L*.
- 2: **if**  $\hat{e}(g^{H_m(csi)} \cdot PK_u, K_p) \stackrel{?}{=} E$  and  $K_c \stackrel{?}{=} \hat{e}(g \cdot PK_u, K_p)$  and  $K_c \notin ST$  **then**
- 3:   **if** *ctr* + 1 > *ε* **then**
- 4:     **return** *False*.
- 5:   **end if**
- 6: **end if**
- 7: Update counter *ctr* = *ctr* + 1 and set *ST* = *ST* ∪ *K<sub>c</sub>*.
- 8: **return** *True*.

(1, 0, ..., 0) for some constance  $\{\omega_i\}_{i \in \mathcal{I}}$ . Then, BC returns  $(\tilde{C}_\delta, \hat{C}_\delta, P_0)$  to DC for user validation. Otherwise, if the smart contract returns *False*, BC returns  $\perp$  to DC and terminates the process.

- **User validation.** When the DC gets  $(\tilde{C}_\delta, \hat{C}_\delta, P_0)$ , he/she computes the following equation:

$$\tilde{C}_\delta^{-1} \stackrel{?}{=} P_0 \cdot \hat{e}(g^{SK_u}, \hat{C}_\delta).$$

If the above equation holds, then the attribute set *S* of the DC is compatible with the hidden access policy (*A*, *ρ*) of ciphertext *CT*. Otherwise, the opposite is true. The DC notifies the BC of the validation result.

**Note:** It is worthy noticing that if and only if the attribute set *S* of the DC matches with the hidden access policy (*A*, *ρ*) of ciphertext *CT*, the decryption testing process of BC will find out the correct subset *I* to compute *P<sub>0</sub>* which is used for user validation by the DC to testify the ciphertext matching result. The motivation of the above designed interaction between BC and DC is that it can not only guarantee the process of decryption testing, but also significantly save the computation cost for DC with resource-limited devices.

- **Outsourced decryption.** If BC is notified that the ciphertext *CT* matches with the request of DC, it forwards the data access request of the DC *ID<sub>u</sub>* to the cloud server of UCP located by *Addr* in order to execute outsourced decryption for the ciphertext *CT* as below:

$$P_1 = \frac{\hat{e}(\hat{C}_1, K)}{\prod_{i \in I} (\hat{e}(C_{1,i}, K') \hat{e}(C_{2,i}, K_{\rho(i)}))^{w_i}} = \hat{e}(g, g)^{\alpha^S} \hat{e}(g, g)^{z_u^S}$$

Finally, the cloud server returns partially decrypted ciphertext (*CT*, *P<sub>0</sub>*, *P<sub>1</sub>*) to DC.

- *Decrypt<sub>U</sub>(PK, CT, P<sub>0</sub>, P<sub>1</sub>, SK<sub>u</sub>)*: Obtaining the ciphertext from UCP off the chain, DC first checks if *I* ∈ *I<sub>A,ρ</sub>*

exists to satisfy  $\{\rho(i)|i \in I\} \subseteq I_s$  and if the following equation holds using system public key *PK* and secret key *SK<sub>u</sub>*, where  $\sum_{i \in I} w_i A_i = (1, 0, \dots, 0)$  holds given the constants  $w_{i \in I}$ .

$$\tilde{C}_\delta^{-1} \stackrel{?}{=} P_0 \cdot \hat{e}(g^{SK_u}, \hat{C}_\delta).$$

If *I* does not exist, DC outputs  $\perp$  as. Otherwise, DC continues the following execution:

$$M = \tilde{C}_1 \cdot \hat{e}(g^{SK_u}, \hat{C}_1) / P_1$$

DC at last obtains the plaintext *M* of ciphertext *CT*.

**Note:** When requesting desired UAV data through BC from UCP which is a DHT-based multi-cloud environment, DC can distributedly access the shared ciphertexts of UAV data. Besides, the data integrity can be ensured by the transactions of BC.

- *UserTrace(PK, DK<sub>u</sub>)* → *ID* or  $\perp$ : If a malicious DC deliberately leaks his decryption key *DK<sub>u</sub>*, any entity can submit to BC a user tracing transaction *T<sub>t</sub>* as below:

$$T_t = (ID_u, TS, Action = \text{“trace malicious user } DK_u'', Sig_t)$$

where *ID<sub>u</sub>* is the identity of any entity in our system. After being verified by BC, *T<sub>t</sub>* triggers user tracing smart contract (abbreviated as UT-SC) to expose the identity of the DC as in Algorithm.3.

**Algorithm 3** Pseudo-Code of *UT – SC(PK, DK<sub>u</sub>)*

**Input:** *PK*: system public key, *DK<sub>u</sub>*: decryption key of malicious DC.

**Output:** *ID* or  $\perp$ .

- 1: Execute **Key Sanity Check**:

$$\hat{e}(g, K) = B \cdot \hat{e}(g^a, (K')^{H_m(K', K'')}) \quad (1)$$

- 2: **if** Eq.1 holds **then**
- 3:   **return** *ID* = *K''*.
- 4: **else**
- 5:   **return**  $\perp$ .
- 6: **end if**

Then, the real identity *ID* of the malicious DC is traced publicly and distributedly by BC.

**Note:** As the user tracing is executed by the smart contract of BC, the proof of the leaked decryption key *DK<sub>u</sub>* cannot be tampered. Thus, it is impossible for the malicious DC to deny his/her abnormal actions.

**C. SECURITY ANALYSIS OF BPADAC**

## 1) SECURITY MODEL

Based on the above system model and formal definition mentioned in Section IV-C, we formalize the security model for BPADAC to depict the ability of adversary. Specifically, we devise a security game between an adversary *A* and a challenger *C* described in Fig.3 to show the

- *Setup*: The challenger  $\mathcal{C}$  initializes the system and generates the system public parameters  $PK$  and master key  $MSK$ . It then sends the  $PK$  to the adversary  $\mathcal{A}$ .
- *Phase 1*: The adversary  $\mathcal{A}$  issues a series of queries  $q_1, \dots, q_n$  adaptively, where  $n$  is a polynomially bounded number and  $q_i$  may be one kind of the following queries:
  - *DKey Query*: The adversary  $\mathcal{A}$  requests his decryption key by submitting an attribute set  $S$  to the challenger  $\mathcal{C}$ . The challenger runs  $KeyGen_U$  and returns the  $DK_u$  to the adversary  $\mathcal{A}$ .
  - *OKey Query*: The adversary  $\mathcal{A}$  launches transformation key queries with an attribute set  $S$  to the challenger  $\mathcal{C}$ . The challenger runs  $KeyGen_O$  and returns the  $TK_u$  to the adversary  $\mathcal{A}$ .
- *Challenge*: The adversary  $\mathcal{A}$  ends the *Phase 1* and submits two equal length messages  $m_0$  and  $m_1$  with two access policy  $\mathcal{A}_0$  and  $\mathcal{A}_1$ . Then, the challenger  $\mathcal{C}$  randomly chooses  $u \in [0, 1]$ , encrypts  $m_u$  and returns the ciphertext  $CT'$  to  $\mathcal{A}$ .
- *Phase 2*: The adversary  $\mathcal{A}$  repeats the *Phase 1* and the queries with submitted attribute set  $S$  that none of the queried attribute sets satisfies  $\mathcal{A}_0$  and  $\mathcal{A}_1$ .
- *Guess*: The adversary  $\mathcal{A}$  outputs a bit  $u' \in [0, 1]$ . If  $u' = u$ ,  $\mathcal{A}$  wins the game. The advantage of  $\mathcal{A}$  is defined as  $Adv_{\mathcal{A}} = |Pr[u' = u] - \frac{1}{2}|$ .

FIGURE 3. Formalized security model of our BPADAC.

indistinguishability of BPADAC against chosen plaintext attacks (IND-CPA). What is more, as we utilize blockchain in our scheme for immutable metadata storage and trustworthy system initialization, we suppose that the immutable property of blockchain is secure which is ensured by blockchain technique and thus can focus on the security model of indistinguishability against chosen plaintext attack for BPADAC.

*Definition 4: The BPADAC scheme is indistinguishable under chosen-plaintext attacks if there exists a probabilistic polynomial time (PPT) adversary  $\mathcal{A}$  that can win the game with non-negligible advantage  $Adv_{\mathcal{A}}$ .*

## 2) SECURITY ANALYSIS

*Theorem 1: Our proposed BPADAC is fully secure under the standard model if the PASH scheme in [19] is fully secure.*

*Proof:* We take the hybrid encryption mechanism to construct our scheme similar to that in PASH [19] such that the security of our BPADAC can be reduced to that of PASH. If the adversary  $\mathcal{A}$  can break the security game mentioned in Section IV-C of our BPADAC with non-negligible advantage  $Adv_0 = \epsilon$ , then a simulator  $\mathcal{B}$  can be constructed to break PASH with advantage  $Adv_1$  which is identical to  $Adv_0$ .  $\square$

- *Setup*: The simulator  $\mathcal{B}$  initializes the PASH scheme and generates the system public parameters  $PK_{PASH} = \{N, g, g^\alpha, H, Y, X_4\}$  and master key  $MSK_{PASH} = \{\alpha, h, X_3\}$ . After getting  $PK_{PASH}$ , the challenger  $\mathcal{C}$  of BPADAC initializes its system public parameters  $PK_{BPADAC} = \{N, g, g^\alpha, \gamma, \theta, g^a, g^b, g^c, H, Y, X_4\}$  and  $MSK_{BPADAC} = \{\alpha, h, X_3\}$ .

- *Phase 1*: The adversary  $\mathcal{A}$  queries decryption key with an attribute set  $S$ . The simulator  $\mathcal{B}$  outputs the decryption key  $DK_u = \{S, K, K', \{K_i\}_{i \in I_S}\}$ , where  $K = g^\alpha g^{at} R$ ,  $K' = g^t R'$ ,  $K_i = (g^{s_i h})^t R_i$ . Then, the challenger  $\mathcal{C}$  randomly picks  $u, u' \in Z_N$ ,  $R_2, R_3 \in G_{p_3}$ , and calculates  $DK_u$  as follows:

$$\begin{aligned} K_{1, \varepsilon_i} &= g^\alpha g^{at} g^{bu} g^{cu' H_1(\varepsilon_i)} R, \\ K_2 &= g^t R_2, K_3 = g^t R_3, K_4 = g^t R', \\ K_{5, x} &= (g^{s_x h})^t R_x, \end{aligned}$$

Then, the challenger  $\mathcal{C}$  returns  $DK_u$  to  $\mathcal{A}$ .

- *Challenge*: The adversary  $\mathcal{A}$  submits two messages  $m_0, m_1$  with equal length and two challenge access policies  $\mathcal{A}_0 = (A, \rho, R_{A0})$ ,  $\mathcal{A}_1 = (A, \rho, R_{A1})$  to the simulator  $\mathcal{B}$ .  $\mathcal{B}$  randomly picks  $u \in [0, 1]$  and outputs the ciphertext  $C = m_u \cdot Y^s$ ,  $C_0 = g^s$ ,  $C_x = g^{aA_x \cdot v} (g^{t\rho(x)} H)^{-r_x} W_{x,1}$ ,  $D_x = g^{r_x} W_{x,2}$ ,  $\bar{C} = Y^{s'}$ ,  $\bar{C}_0 = g^{s'} \bar{W}$ ,  $\bar{C}_x = g^{aA_x \cdot v'} (g^{t\rho(x)} H)^{-s'} \bar{W}_x$  with access policy  $\mathcal{A}_u$ . Then,  $\mathcal{B}$  computes the following  $CT_{\mathcal{A}_u}$ :

$$\begin{aligned} C_{2, \varepsilon_i} &= (g_1^{cH_1(\varepsilon_i)})^s H_{2, \varepsilon_i}, C_{3, x} = C_x, C_{4, x} = D_x, \\ \bar{C} &= \bar{C}, \bar{C}_0 = \bar{C}_0, \bar{C}_1 = (\bar{C}_0)^b \bar{H}_1, \\ \bar{C}_{2, \varepsilon_i} &= (\bar{C}_0)^{cH_1(\varepsilon_i)} \bar{H}_{2, \varepsilon_i}, \bar{C}_{3, x} = \bar{C}_x \bar{H}_{3, x}, \end{aligned}$$

- *Phase 2*: The adversary  $\mathcal{A}$  repeats the queries as *Phase 1* and none of the queried attribute sets satisfies  $\mathcal{A}_0, \mathcal{A}_1$ .
- *Guess*: The adversary  $\mathcal{A}$  guess the  $u'$  and forwards it to the challenger  $\mathcal{C}$  through the simulator  $\mathcal{B}$ . If  $u' = u$ , the adversary  $\mathcal{A}$  wins the security game of our BPADAC,

TABLE 3. Computation comparison in various schemes.

Schemes	KeyGen	UserEnc	UserDec	UserTrace
PASH	$(2 S  + 3)E$	$(7l + 4)E + 2E_T$	$2 I E + ( I  + 1)E_T$ $+ (2 I  + 3)P$	–
HTAC	$(2 S  + 4)E$	$(7l + 5)E + 2E_T$	$(3 I  + 4)E + ( I  + 1)E_T$ $+ (2 I  + 4)P$	$(2 S  + 2)E$ $+ (2 S  + 5)P$
PH-LU-CPABE	$(2 S  + 3)E$	$(6l + 2)E + 2E_T$	$(4 I  + 4)E + E_T + 4P$	$E + 2P$
BPADAC	$(2 S  + 3)E$	$2 S E$	$2E$	$E + 2P$

Note. " $l$ " is the row number in access policy, " $|I|$ " is the complexity of access policy in decryption.

TABLE 4. Storage comparison in various schemes.

Schemes	PPSize	DKSize	CTSize	TraceSize
PASH	$4 G_{p_i}  +  G_T $	$( S  + 2) G_{p_i p_j} $	$(2l + 3) G_{p_i p_j}  + 2 G_T $	–
HTAC	$5 G_{p_i}  +  G_T $	$( S  + 3) G_{p_i p_j}  +  Z_N $	$(3l + 4) G_{p_i p_j}  + 2 G_T $	$ L  Z_N $
PH-LU-CPABE	$4 G_{p_i}  +  G_T $	$( S  + 2) G_{p_i p_j}  +  Z_N $	$(2l + 2) G_{p_i p_j}  + 2 G_T $	$\odot$
BPADAC	$5 G_{p_i}  +  G_T $	$( S  + 2) G_{p_i p_j}  +  Z_N $	$(2l + 3) G_{p_i p_j} $ $+ 2 G_T  + 3l Z_N $	$\odot$

Note. " $|L|$ " is the size of user table for tracing, " $\odot$ " is the efficient symbol.

then the challenger  $\mathcal{C}$  can break the security game of PASH scheme and the security of our scheme reduces to that of PASH. The advantage of  $\mathcal{A}$  to break our scheme is identical to that of  $\mathcal{C}$  to break PASH.

As a summary, if PASH is fully secure, our BPADAC is fully secure in standard model.

## VI. PERFORMANCE ANALYSIS

This section presents the analysis of the efficiency for our BPADAC including both theoretical analysis and experimental performance analysis. In these analysis, we compare our scheme and other three excellent existing schemes, that is, PASH [19], HTAC [21] and PH-LU-CPABE [20].

### A. THEORETICAL ANALYSIS

In theoretical analysis of BPADAC, we compare our scheme and other three related existing schemes from view of computation complexity and storage complexity. In our analysis,  $E$  and  $E_T$  denotes exponentiation in group  $G$  and  $G_T$ ,  $P$  denotes pairing operation in  $\hat{e}$ ,  $|S|$  denotes the number of access attribute set  $S$ ,  $|G_{p_i}|$ ,  $|G_{p_i p_j}|$ ,  $|G_T|$  and  $|Z_N|$  denote the length of element of  $G_{p_i}$ ,  $G_{p_i} \cdot G_{p_j}$ ,  $G_T$  and  $Z_N$ , respectively.

Table 3 shows the comparison results of the computation complexity in the above four schemes from aspects of time cost in *KeyGen*, *UserEnc*, *UserDec*, *UserTrace* algorithms. Firstly, it is apparent that in *KeyGen* algorithm, BPADAC costs the same computation as PASH and PH-LU-CPABE but less computation than HTAC which relies on TA for centralized user tracing and incurs in more key components generation in *KeyGen* algorithm. Secondly, the computation cost of *UserEnc* algorithm in BPADAC that evaluate the complexity of final data encryption is far less than that of PASH, HTAC and PH-LU-CPABE due to the employment

of online/offline encryption for computation task offload and complexity mitigation. Thirdly, in *UserDec* algorithm, the computation complexity of BPADAC is significantly less than other schemes because BPADAC just executes two exponential operations in  $G$  as the result of outsourced decryption. Besides, in *UserTrace* algorithm, as PASH fails to implement user tracing, we only compare our BPADAC with HTAC and PH-LU-CPABE schemes. Apparently, BPADAC costs the same computation as PH-LU-CPABE and far less computation than HTAC that needs extra computation in key sanity check.

Table 4 gives the comparison in storage complexity for the above four schemes from aspects of the storage overhead in *Setup*, *KeyGen*, *UserEnc*, *UserTrace*. From the view of storage overhead evaluation in *Setup* algorithm, i.e., the storage complexity of public parameters, it is obvious that our BPADAC has the same storage complexity for public parameters as HTAC. However, PASH and PH-LU-CPABE costs less storage. The reason for this is that BPADAC and HTAC cost one more element of  $G$  for public parameters in *Setup* algorithm used in outsourced decryption in standard model. With respect to the storage overhead of user decryption key in *KeyGen* algorithm, BPADAC costs the same storage for user decryption key generation as PH-LU-CPABE and less of that than HTAC which relies on centralized TA for user tracing and thus generates more components in user decryption key. Moreover, in *UserEnc* algorithm, BPADAC costs slightly more storage in ciphertext than the other schemes because it has to generate other three elements of  $Z_N$  in ciphertext as a result of the introduction of online/offline encryption. What is more, from the point of view of *UserTrace* algorithm, the storage overhead for user tracing in BPADAC and PH-LU-CPABE is much more advantageous than HTAC because both of the two schemes

achieve efficient public user tracing without the maintenance of a user list as in HTAC.

### B. EXPERIMENTAL ANALYSIS

For experimental analysis, we implement BPADAC, PASH, HTAC and PH-LU-CPABE by developing in Java programming language above JDK64-8 with type A curve over an elliptic curve group of Java Pairing-Based Cryptography library (JPBC) [65], i.e., a capsulation of pairing-based cryptography library [66] in Java. Then, we deploy these implementations in a server equipped with Intel Core i5-6500 3.20GHz, 8GB memory and windows server 2019 operating system to simulate UCP, a HuaweiP20 smartphone with Android OS 6.0 operation system to simulate DP as well as several drones as UAV devices. Besides, we utilize Ethereum to implement the blockchain platform and transactions in our experiments with smart contracts developed in Solidity programming language and deployed on Ethereum. As our experiments are running in blockchain test net of Ethereum, the ether worth no real value and the consumption is measured by the unit Gas.

In the following experiments, attribute universe size is ranging in  $|U| \in [5, 50]$  for public parameter storage cost evaluation. User attribute set  $|S|$  is set in size of  $[5, 50]$  to evaluate time and storage costs in key generation. To evaluate the time and storage cost in encryption,  $l$  (row number of access policy) is ranging from 10 to 50. We set the complexity of access policy  $|I|$  in  $[5, 10, 15, 20]$  with the number of files ranging in  $[5, 10, 15, 20]$  to assess the time cost in decryption. Fig.4 summarizes the experimental performance comparison.

Fig.4(a) and Fig.4(b) give the time and storage cost in key generation algorithm of user registration phase. From the comparison of our BPADAC and other three related schemes, we notice that BPADAC costs similar time in key generation as PASH, HTAC and PH-LU-CPABE. However, BPADAC and HTAC bring about a bit more overhead in user decryption key storage compared with PASH and PH-LU-CPABE. From this point of view, we can also evaluate the effect of user tracing in key generation, that is, to realize user tracing with white-box mechanism, BPADAC and HTAC need to generate more components in decryption key than other two schemes. This is the right reason of why the key storage of BPADAC and HTAC is slightly more than that of the other two schemes.

Fig.4(c) and Fig.4(d) depicts the time cost and storage overhead of data encryption algorithm in data outsourcing phase. Obviously, BPADAC costs much less time than PASH, HTAC and PH-LU-CPABE in data encryption as shown in Fig.4(c). The reason is that BPADAC utilizes online/offline technique in encryption that offloads part of encryption computation to offline phase in advance such that the time cost of actual encryption (i.e., online encryption) is much less than that of other schemes. Nevertheless, as shown in Fig.4(d), BPADAC costs slightly more in ciphertext storage due to the introduction of online/offline encryption that brings about three more elements in  $Z_N$  for each line of LSSS

TABLE 5. The gas cost of BPADAC.

Operation	Size(bytes)	GAS	Ether
Encryption	1804	127491	0.00025498
Decryption	574	50111	0.00010022
UserTrace	410	35793	0.00007159

policy than the other schemes. However, the ciphertexts are outsourced to distributed clouds, which will not affect the resource-limited UAV devices. Thus, the comprehensive evaluation of time cost and storage overhead in encryption is a rational tradeoff between time and space complexity of encryption algorithm. The result is that our BPADAC outperforms other three schemes from the view of data encryption.

Fig.4(e) and Fig.4(f) plot the comparison of the four schemes from the view of the time cost of user decryption algorithm and the storage cost of system public parameters, respectively. It is worthy noticing that from Fig.4(e), we can conclude that BPADAC costs much less time in user decryption than the other three schemes. The reason is the utilization of outsourced decryption in BPADAC with both decryption testing and ciphertext decryption outsourced to distributed clouds, which saves much time cost of user decryption for BPADAC. In the evaluation for the storage cost of public parameters in Fig.4(f), BPADAC has the same storage cost for public parameters as that of PASH and PH-LU-CPABE. Whereas, the storage cost of public parameters in HTAC is much more than other schemes because it leverages centralized white-box user tracing mechanism which incurs in extra components in public parameters for user traceability implementation.

Finally, we evaluate the cost of Ethereum blockchain platform in different phases of our BPADAC, and give a summary of its gas consumption in Table 5. In this experimental evaluation, we set the UAV data file number as 1 and the size of user (including DP in encryption and DC in decryption) attribute set  $|S|$  is set as 5 while the line number of access policy matrix of ciphertext is set as  $l = 20$ . In encryption phase, we only store part of the policy-related ciphertext components to blockchain to improve storage and computation efficiency for miners of blockchain, while in decryption, these metadata are used in outsourced decryption testing with the transformation key of DC in corresponding transaction. In UserTrace operation, the blockchain needs to store the disclosed decryption key to trace the malicious user and record the proof of abnormal behavior. Thus, we evaluate the corresponding cost in Ethereum blockchain in the aforementioned three phases.

As a summary, from view of theoretical and experimental performance analysis, BPADAC performs consistent in each aspects. Moreover, we can obviously conclude that BPADAC is advantageous in both encryption and decryption. This characteristic makes it more compatible with resource-limited

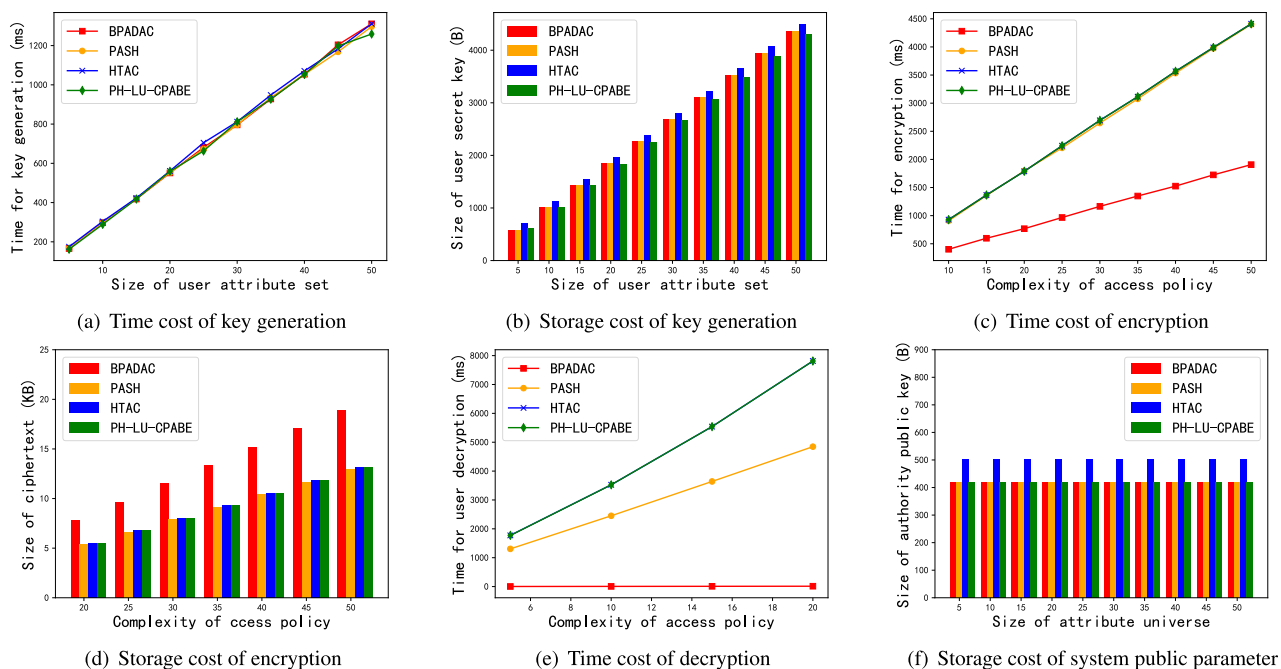


FIGURE 4. Performance evaluation of BPADAC.

end devices, such as UAV devices in our schemes. The only drawback of BPADAC is the storage cost of ciphertext. Nevertheless, we introduce the architecture of multi-cloud in UCP of our scheme, which means UCP can provide unlimited resources. Furthermore, BPADAC can provide the functionalities of access time limitation, distributed data access and undeniable and public user tracing besides large attribute universe, expressive access policy and policy privacy preservation. Therefore, BPADAC is more practical for UAV data sharing in cloud-based IoD systems.

### VII. CONCLUSION

In this paper, after deeply analyzing the problems of UAV data sharing in cloud-based IoD systems, we proposed a blockchain-based privacy-aware data access control (BPADAC) scheme for distributed and secure UAV data sharing in such a mobile and distributed large-scale environment and gave its formal models and definitions with detailed constructions. With the help of blockchain and CP-ABE techniques, BPADAC achieves fine-grained and distributed data access such that any privileged data user has the ability to access UAV data through blockchain. In the meantime, the UAV data sharing service can be guaranteed through the mechanism of limited access times. Moreover, multi-cloud combining with DHT technique can store large-scale UAV data in a distributed and scalable way, and eliminate the drawbacks of traditional centralized cloud. Partial policy hiding is employed by BPADAC to provide access policy privacy protection for outsourced UAV data in clouds. Furthermore, BPADAC is able to deal with traitor tracing efficiently and publicly by utilizing public user tracing approach without any denial. In addition, the security and performance analysis with a prototype implemented on

Ethereum blockchain provide strong evidences that BPADAC is secure and suitable for UAV data sharing in cloud-based IoD systems. Our future work will be devoted to study the identification problems of UAV data source and outsourced UAV data in cloud-based IoD systems.

### ACKNOWLEDGMENT

An earlier version of this paper was presented in part at the 4th EAI International Conference on Security and Privacy in New Computing Environments (EAI SPNCE 2021) [DOI: 10.1007/978-3-030-96791-8\_8].

### REFERENCES

- [1] X. Li, H. Liu, W. Wang, Y. Zheng, H. Lv, and Z. Lv, "Big data analysis of the Internet of Things in the digital twins of smart city based on deep learning," *Future Gener. Comput. Syst.*, vol. 128, pp. 167–177, Mar. 2022.
- [2] F. Tang, X. Chen, M. Zhao, and N. Kato, "The roadmap of communication and networking in 6G for the metaverse," *IEEE Wireless Commun.*, early access, Jun. 24, 2022, doi: 10.1109/MWC.019.2100721.
- [3] M. A. Khan, N. Kumar, S. A. H. Mohsan, W. U. Khan, M. M. Nasralla, M. H. Alsharif, J. Zywiolok, and I. Ullah, "Swarm of UAVs for network management in 6G: A technical review," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 1, pp. 741–761, Mar. 2023.
- [4] Z. Na, C. Ji, B. Lin, and N. Zhang, "Joint optimization of trajectory and resource allocation in secure UAV relaying communications for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16284–16296, Sep. 2022.
- [5] S. Yu, A. K. Das, Y. Park, and P. Lorenz, "SLAP-IoD: Secure and lightweight authentication protocol using physical unclonable functions for Internet of Drones in smart city environments," *IEEE Trans. Veh. Technol.*, vol. 71, no. 10, pp. 10374–10388, Oct. 2022.
- [6] W. Wang, T. Chen, R. Ding, G. Seco-Granados, L. You, and X. Gao, "Location-based timing advance estimation for 5G integrated LEO satellite communications," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6002–6017, Jun. 2021.
- [7] H. Xu, Z. Chen, H. Liu, L. Chang, T. Huang, S. Ye, L. Zhang, and C. Du, "Single-fed dual-circularly polarized stacked dielectric resonator antenna for K/Ka-band UAV satellite communications," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4449–4453, Apr. 2022.

- [8] K.-Y. Tsao, T. Girdler, and V. G. Vassilakis, "A survey of cyber security threats and solutions for UAV communications and flying ad-hoc networks," *Ad Hoc Netw.*, vol. 133, Aug. 2022, Art. no. 102894.
- [9] Y. Dang, C. Benzaid, B. Yang, T. Taleb, and Y. Shen, "Deep-ensemble-learning-based GPS spoofing detection for cellular-connected UAVs," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25068–25085, Dec. 2022.
- [10] J. Zhang, J. Ma, T. Li, and Q. Jiang, "Efficient hierarchical and time-sensitive data sharing with user revocation in mobile crowdsensing," *Secur. Commun. Netw.*, vol. 2021, pp. 1–17, Feb. 2021.
- [11] J. Zhang, T. Li, M. S. Obaidat, C. Lin, and J. Ma, "Enabling efficient data sharing with auditable user revocation for IoV systems," *IEEE Syst. J.*, vol. 16, no. 1, pp. 1355–1366, Mar. 2022.
- [12] S. Das and S. Namasudra, "Multiauthority CP-ABE-based access control model for IoT-enabled healthcare infrastructure," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 821–829, Jan. 2023.
- [13] Q. Tong, Y. Miao, H. Li, X. Liu, and R. Deng, "Privacy-preserving ranked spatial keyword query in mobile cloud-assisted fog computing," *IEEE Trans. Mobile Comput.*, early access, Dec. 13, 2022, doi: [10.1109/TMC.2021.3134711](https://doi.org/10.1109/TMC.2021.3134711).
- [14] Y. Li, J. Ma, Y. Miao, Y. Wang, T. Yang, X. Liu, and K.-K.-R. Choo, "Traceable and controllable encrypted cloud image search in multi-user settings," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2936–2948, Oct. 2022.
- [15] Y. Guo, Z. Lu, H. Ge, and J. Li, "Revocable blockchain-aided attribute-based encryption with escrow-free in cloud storage," *IEEE Trans. Comput.*, early access, Jan. 5, 2023, doi: [10.1109/TC.2023.3234210](https://doi.org/10.1109/TC.2023.3234210).
- [16] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 762–773, Apr. 2022.
- [17] N. Chen, J. Li, Y. Zhang, and Y. Guo, "Efficient CP-ABE scheme with shared decryption in cloud storage," *IEEE Trans. Comput.*, vol. 71, no. 1, pp. 175–184, Jan. 2022.
- [18] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 785–796, Jan. 2016.
- [19] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: Efficient policy-hiding attribute-based access control," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2130–2145, Jun. 2018.
- [20] P. Zeng, Z. Zhang, R. Lu, and K.-K.-R. Choo, "Efficient policy-hiding and large universe attribute-based encryption with public traceability for Internet of Medical Things," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10963–10972, Jul. 2021.
- [21] Q. Li, Y. Zhang, T. Zhang, H. Huang, Y. He, and J. Xiong, "HTAC: Fine-grained policy-hiding and traceable access control in mHealth," *IEEE Access*, vol. 8, pp. 123430–123439, 2020.
- [22] Y. Yang, J. Zhang, X. Liu, Q. Jiang, Y. Liu, B. Li, Y. Du, and J. Ma, "A scalable and auditable secure data sharing scheme with traceability for fog-based smart logistics," *IEEE Internet Things J.*, early access, Nov. 9, 2022, doi: [10.1109/IJOT.2022.3220850](https://doi.org/10.1109/IJOT.2022.3220850).
- [23] Z. Guo, G. Wang, Y. Li, J. Ni, R. Du, and M. Wang, "Accountable attribute-based data-sharing scheme based on blockchain for vehicular ad hoc network," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 7011–7026, Apr. 2023, doi: [10.1109/IJOT.2022.3228550](https://doi.org/10.1109/IJOT.2022.3228550).
- [24] Y. Yang, R.-H. Shi, K. Li, Z. Wu, and S. Wang, "Multiple access control scheme for EHRs combining edge computing with smart contracts," *Future Gener. Comput. Syst.*, vol. 129, pp. 453–463, Apr. 2022.
- [25] L. Cheng and F. Meng, "An improvement on 'CryptCloud<sup>+</sup>': Secure and expressive data access control for cloud storage," *IEEE Trans. Services Comput.*, early access, Sep. 27, 2022, doi: [10.1109/TSC.2022.3210114](https://doi.org/10.1109/TSC.2022.3210114).
- [26] H. Arshad, C. Johansen, O. Owe, P. Picazo-Sanchez, and G. Schneider, "Semantic attribute-based encryption: A framework for combining ABE schemes with semantic technologies," *Inf. Sci.*, vol. 616, pp. 558–576, Nov. 2022.
- [27] H. Nasirae, M. Ashouri-Talouki, and X. Liu, "Optimal black-box traceability in decentralized attribute-based encryption," *IEEE Trans. Cloud Comput.*, early access, Sep. 28, 2022, doi: [10.1109/TCC.2022.3210137](https://doi.org/10.1109/TCC.2022.3210137).
- [28] T. Li, J. Zhang, M. S. Obaidat, C. Lin, Y. Lin, Y. Shen, and J. Ma, "Energy-efficient and secure communication toward UAV networks," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 10061–10076, Jun. 2022, doi: [10.1109/IJOT.2021.3118079](https://doi.org/10.1109/IJOT.2021.3118079).
- [29] J. Zhou, Z. Cao, Z. Qin, X. Dong, and K. Ren, "LPPA: Lightweight privacy-preserving authentication from efficient multi-key secure outsourced computation for location-based services in VANETs," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 420–434, 2019.
- [30] J. Hong, K. Xue, N. Gai, D. S. L. Wei, and P. Hong, "Service outsourcing in F2C architecture with attribute-based anonymous access control and bounded service number," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 5, pp. 1051–1062, Sep. 2020.
- [31] T. H. Yuen, J. K. Liu, M. H. Au, X. Huang, W. Susilo, and J. Zhou, "k-times attribute-based anonymous access control for cloud computing," *IEEE Trans. Comput.*, vol. 9, no. 64, pp. 2595–2608, Sep. 2015.
- [32] J. Zhang, Y. Yang, N. Lu, Z. Liu, and J. Ma, "A privacy-aware and time-limited data access control scheme with large universe and public traceability for cloud-based IoD," in *Proc. Int. Conf. Secur. Privacy New Comput. Environ.* Cham, Switzerland: Springer, 2021, pp. 103–116.
- [33] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 762–771, Sep./Oct. 2019.
- [34] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.
- [35] Y. Miao, Q. Tong, K.-K.-R. Choo, X. Liu, R. H. Deng, and H. Li, "Secure online/offline data sharing framework for cloud-assisted industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8681–8691, Oct. 2019.
- [36] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Proc. Int. Workshop Public Key Cryptogr.* Cham, Switzerland: Springer, 2014, pp. 293–310.
- [37] Z. Zhang, P. Zeng, B. Pan, and K.-K.-R. Choo, "Large-universe attribute-based encryption with public traceability for cloud storage," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10314–10323, Oct. 2020.
- [38] K. Zhang, H. Li, J. Ma, and X. Liu, "Efficient large-universe multi-authority ciphertext-policy attribute-based encryption with white-box traceability," *Sci. China Inf. Sci.*, vol. 61, no. 3, pp. 1–13, Mar. 2018.
- [39] J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proc. 7th ACM Symp. Inf., Comput. Commun. Secur.*, 2012, pp. 18–19.
- [40] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected UAVs: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2125–2140, Apr. 2019.
- [41] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.
- [42] Z. Ye, K. Wang, Y. Chen, X. Jiang, and G. Song, "Multi-UAV navigation for partially observable communication coverage by graph reinforcement learning," *IEEE Trans. Mobile Comput.*, early access, Jan. 31, 2022, doi: [10.1109/TMC.2022.3146881](https://doi.org/10.1109/TMC.2022.3146881).
- [43] Q. Zhang, J. Chen, L. Ji, Z. Feng, Z. Han, and Z. Chen, "Response delay optimization in mobile edge computing enabled UAV swarm," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3280–3295, Mar. 2020.
- [44] X. You, C.-X. Wang, J. Huang, X. Gao, Z. Zhang, M. Wang, Y. Huang, C. Zhang, Y. Jiang, and J. Wang, "Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *Sci. China Inf. Sci.*, vol. 64, no. 1, pp. 1–74, Nov. 2020.
- [45] N. Pathak, S. Misra, A. Mukherjee, A. Roy, and A. Y. Zomaya, "UAV virtualization for enabling heterogeneous and persistent UAV-as-a-service," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6731–6738, Jun. 2020.
- [46] T. Alladi, N. Naren, G. Bansal, V. Chamola, and M. Guizani, "SecAuthUAV: A novel authentication scheme for UAV-ground station and UAV-UAV communication," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15068–15077, Dec. 2020.
- [47] P. Mehta, R. Gupta, and S. Tanwar, "Blockchain envisioned UAV networks: Challenges, solutions, and comparisons," *Comput. Commun.*, vol. 151, pp. 518–538, Feb. 2020.
- [48] Z. Liu, S. Duan, P. Zhou, and B. Wang, "Traceable-then-revocable ciphertext-policy attribute-based encryption scheme," *Future Gener. Comput. Syst.*, vol. 93, pp. 903–913, Oct. 2019.
- [49] J. Ning, X. Dong, Z. Cao, and L. Wei, "Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2015, pp. 270–289.
- [50] Z. Liu, J. Xu, Y. Liu, and B. Wang, "Updatable ciphertext-policy attribute-based encryption scheme with traceability and revocability," *IEEE Access*, vol. 7, pp. 66832–66844, 2019.

- [51] K. Xue, Y. Xue, J. Hong, W. Li, H. Yue, D. S. Wei, and P. Hong, "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 953–967, Apr. 2017.
- [52] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. USENIX Secur. Symp.*, 2011, pp. 1–16.
- [53] S. J. De and S. Ruj, "Efficient decentralized attribute based access control for mobile clouds," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 124–137, Jan. 2020.
- [54] J. Lai, R. H. Deng, and Y. Li, "Fully secure ciphertext-policy hiding CP-ABE," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.* Guangzhou, China: Springer, May 2011, pp. 24–39.
- [55] Y. Ren, Y. Leng, J. Qi, P. K. Sharma, J. Wang, Z. Almkhadme, and A. Tolba, "Multiple cloud storage mechanism based on blockchain in smart homes," *Future Gener. Comput. Syst.*, vol. 115, pp. 304–313, Feb. 2021.
- [56] S. Roy, A. K. Das, S. Chatterjee, N. Kumar, S. Chattopadhyay, and J. J. Rodrigues, "Provably secure fine-grained data access control over multiple cloud servers in mobile cloud computing based healthcare applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 457–468, Jan. 2019.
- [57] F. Li, K. Liu, L. Zhang, S. Huang, and Q. Wu, "EHRChain: A blockchain-based EHR system using attribute-based and homomorphic cryptosystem," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2755–2765, Sep. 2022.
- [58] C. Feng, K. Yu, A. K. Bashir, Y. D. Al-Otaibi, Y. Lu, S. Chen, and D. Zhang, "Efficient and secure data sharing for 5G flying drones: A blockchain-enabled approach," *IEEE Netw.*, vol. 35, no. 1, pp. 130–137, Jan. 2021.
- [59] W. Liang, Y. Yang, C. Yang, Y. Hu, S. Xie, K.-C. Li, and J. Cao, "PDPChain: A consortium blockchain-based privacy protection scheme for personal data," *IEEE Trans. Rel.*, early access, Aug. 5, 2022, doi: [10.1109/TR.2022.3190932](https://doi.org/10.1109/TR.2022.3190932).
- [60] L. Zhang, T. Zhang, Q. Wu, Y. Mu, and F. Rezaeibagha, "Secure decentralized attribute-based sharing of personal health records with blockchain," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12482–12496, Jul. 2022.
- [61] Z. Zhang, J. Zhang, Y. Yuan, and Z. Li, "An expressive fully policy-hidden ciphertext policy attribute-based encryption scheme with credible verification based on blockchain," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8681–8692, Jun. 2022.
- [62] Y. Jiang, X. Xu, and F. Xiao, "Attribute-based encryption with blockchain protection scheme for electronic health records," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 3884–3895, Dec. 2022.
- [63] Y. He, H. Wang, Y. Li, K. Huang, V. C. M. Leung, F. R. Yu, and Z. Ming, "An efficient ciphertext-policy attribute-based encryption scheme supporting collaborative decryption with blockchain," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2722–2733, Feb. 2022.
- [64] L. Zhang, Y. Zhang, Q. Wu, Y. Mu, and F. Rezaeibagha, "A secure and efficient decentralized access control scheme based on blockchain for vehicular social networks," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17938–17952, Sep. 2022.
- [65] A. De Caro and V. Iovino, "JPBC: Java pairing based cryptography," in *Proc. 16th IEEE Symp. Comput. Commun. (ISCC)*, Kerkyra, Greece, Jun. 2011, pp. 850–855.
- [66] A. Riyal, G. Kumar, and D. K. Sharma, "Pairing-based cryptography," in *Functional Encryption*. Berlin, Germany: Springer, 2021, pp. 79–101.



**ZHUO MA** received the B.S., M.S., and Ph.D. degrees from the School of Telecommunications Engineering, Xidian University, China, in 2003, 2006, and 2012, respectively. He is currently a Lecturer in telecommunications engineering with Xidian University. His research interests include the fusion application of UAV/UGV, satellite communication, and machine learning in industrial internet area.



**JIawei ZHANG** received the B.S. and M.S. degrees from the School of Telecommunications Engineering, Xidian University, China, in 2007 and 2010, respectively, and the Ph.D. degree from the School of Computer Science and Technology, Xidian University, in 2021. His current research interests include access control, data security, cloud and edge security, blockchain, cryptography, and network security.

...