A Distributed Publisher-Driven Secure Data Sharing Scheme for Information-Centric IoT

Dr.K.Srinivas Babu1 C.Swathi2
1Professor, Department of Computer Science and Engineering, 2M.Tech

Abstract—In Information-Centric Internet of Things (I-CIoT), IoT data can be cached throughout a network for close data copy retrievals. Such a distributed data caching environment, however, poses a challenge to flexible authorization in the network. To address this challenge, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) has been identified as a promising approach. However, in the existing CP-ABE scheme, publishers need to retrieve attributes from a centralized server for encrypting data, which leads to high communication overhead. To solve this problem, we incorporate CP-ABE and propose a novel Distributed Publisher-driven secure Data sharing for ICIoT (DPD-ICIoT) to enable only authorized users to retrieve IoT data from distributed cache. In DPD-ICIoT, newly introduced Attribute Manifest (AM) is cached in the network, through which publishers can retrieve the attributes from nearby copy holders instead of a centralized attribute server. In addition, a key chain mechanism is utilized for efficient cryptographic operations, and an Automatic Attribute Self-update Mechanism (AASM) is proposed to enable fast updates of attributes without querying centralized servers. According to the performance evaluation, DPD-ICIoT achieves lower bandwidth cost compared to the existing CP-ABE scheme.

Keywords—IoT, ICN, Security.

I. INTRODUCTION

IoT (Internet of Things) is likely to have a major impact on human lives as new services and applications are developed through integration of the physical and digital worlds [1][2]. It is predicted that 50 billion devices will be connect-ed through IoT by 2020, and vast amounts of data will be generated from those devices [4]. Today, most IoT services are designed based on Internet technology [2], which was originally conceived for end-to-end communications. Based on such technology, IoT data sharing applications have been developed on the basis of centralized servers/clouds, which produce redundant and duplicate traffic and bring out large latencies. Such a considerable volume of redundant traffic hinders efficient data flows and impose limitations on providing highly available services as is required by IoT applications [5].

With regard to the use of IoT applications, users are usually concerned only about the IoT data that they retrieve rather than where the data are stored or cached [6][7]. Information-Centric Networking (ICN) is an emerging technology that enables users to retrieve data from close caches without the need to access distant servers or clouds each time [6][8][9]. Reducing the redundant traffic overhead and data retrieval latency by moving data from clouds to caches close to users is a promising approach. It integrates computing power and storage to alleviate the bottleneck of network bandwidth resources [5]. Among the existing IC-Ns, Content-Centric Network (CCN)/Named Data Network (NDN) [6][9] is one of the most promising architectures; therefore, in this paper, we focus on CCN/NDN.

Compared to Internet-based IoT designs, ICN-based IoT designs have several salient and distinctive features with regard to security, heterogeneity, fast configuration, and diverse communication paradigms [10-16][19], besides a reduction in traffic and latency. ICN is expected to be one of the fundamental technologies that will support IoT applications and services in the future, and for simplicity, hereafter, we refer to the IoT designs using ICN as ICIoT. ICIoTs have recently been widely discussed for use in IoT applications, such as smart cities [10], smart grid [16], smart home [14], IoT data sharing [11], service-oriented architectures [13], and data collection in IoT [15]. The design requirements and challenges as well as the applicability of ICIoT have also been discussed in IRTF ICN RG [12]. ICIoTs has emerged as a promising solution to provide viable IoT services to users [11][12][19].

To realize a true IoT vision, ensuring security is a key issue [1-3][29][32-36]. Along these lines, some of the primary security threats that IoT data sharing tends to face include unauthorized access, illegal modification, and impersonated publication and retrieval. It is necessary to design a flexible and secure IoT data sharing scheme, wherein IoT data are securely published, cached in the network, and retrieved by only authorized users. However, because of unpredictable caching of IoT data on untrusted devices as is typical in ICIoTs, it is challenging to provide fine-grained data access control in a
distributed caching environment to future IoT services.

Conventional work on IoT network security and AAA is designed based on an end-to-end principle, which is not adequate for emerging ICIoT. Security for ICIoT, in particular, has also been recently discussed and investigated in the literature [17], such as key management for information-centric smart metering infrastructure [18], and securing building management system using named data approach [19].

However, none of them focus on flexible IoT data access control in ICIoT. Conversely, flexible and fine-grained access control in cloud computing [39] has been realized through Ciphertext Policy Attribute-Based Encryption (CP-ABE) [21] with centralized server(s), wherein all attribute values and access policies are retrieved from the servers. In this paper, we denote these schemes as the existing CP-ABE scheme. The existing CP-ABE scheme does not consider a ubiquitous distributed caching environment and completely relies on centralized servers/clouds, which restricts the scalability of IoT systems.

To address this issue, we propose a Distributed Publisher-driven secure Data sharing for ICIoT (DPD-ICIoT) to enable IoT data to be securely shared based on publisher-defined policy. DPD-ICIoT provides flexible authorization from publishers to users. In DPD-ICIoT, CP-ABE is employed to provide flexible authorization from publishers to users. To balance centralized management and distributed retrievals for attributes, attribute manifest (AM) and data manifest (DM) are introduced and distributedly cached in the network. Thus, publishers can retrieve AMs from close copy holders instead of the centralized attribute servers. Herein, AM and DM are the data chunks, with the type of “Manifest”, that describe attributes and data, respectively. Further, to reduce the large traffic overhead of attribute updates, we propose an Automatic Attribute Self-update Mechanism (AASM) to enable the update of attributes without querying the distant server. Compared with the existing CP-ABE scheme, the total bandwidth cost in packet transmissions consumed for attribute retrievals can be greatly reduced.

The main contributions of this paper are as follows. (1) To the best of our knowledge, this is the first work to investigate publisher-driven fine-grained access control in a ubiquitously distributed caching scenario for ICIoT. We integrate CP-ABE with the typical ICN, CCN/NDN [6][9], and propose a novel DPD-ICIoT scheme for providing distributed, secure, and flexible data sharing for ICIoT. (2) We employ a key chain mechanism for efficient data encryption and decryption. (3) We design the AM to enable the close copy retrievals of attributes and propose an AASM to provide efficient attribute update. (4) System evaluation is performed to compare the proposed DPD-ICIoT scheme with the existing CP-ABE scheme.

The remainder of this paper is organized as follows. Section II provides the system description. The threats and security requirements are given in Section III. Section IV introduces the proposed DPD-ICIoT scheme, where the building block and AASM are proposed. The security analysis and characteristics of the proposed DPD-ICIoT scheme are provided in Section V. Section VI provides system evaluations. The paper is concluded in Section VII.

II. SYSTEM DESCRIPTION

To provide IoT services based on Internet technology, central servers/clouds are typically deployed for storing the data collected from IoT devices. However, this paradigm results in large latencies and much traffic overhead because of the considerable number of duplicate data retrievals from distant servers/clouds. On the other hand, routers are expected to be equipped with caches. It can be predicted that IoT data move from centralized servers/clouds to the edge of a network, such as caches surrounding users [5].

Consider a system in ICIoT. IoT data are cached in a distributed manner in the network after they are published by publishers. Then, users retrieve them from close copy holders [40]. After data are published, publishers lose control over the data, and therefore, it would be challenging to make the data only accessible based on a publisher-defined access policy, while also inhibiting attacks, such as unauthorized access, illegal modification, and impersonation attack.

Herein, we envisage a typical IoT use scenario for such ICIoT system as in Fig. 1, where IoT data are published by publishers, cached throughout the network, and retrieved by users from the caches.

In the scenario, besides the physical entities for communication, the entities that logically play roles in IoT data disseminations are as follows.

- **User**: The entity who retrieves data from server(s) or caches in the network.
  - **Publisher**: The entity who publishes IoT data targeted for a set of Users.
• NOA (Network Operator and Authority): The entity who operates a network consisting of routers, gateways, and access points, which are potentially equipped with caches. It provides security policies and functions for the devices in the network, such as functions for identity management and authentication services for entities.

• DSA (Data Sharing Authority): The entity that assists Publishers to provide access privileges to Users for securely providing their IoT data.

In Fig. 1, there are three administrative domains, Domaina, Domainb, and Domainc, serving three areas. An administrative domain in this paper is a group of network devices, such as routers, base stations (BSs), gateways, access points, and links among them, which have a common security policy and configurations. A domain identifies the boundary for security settings, and different domains may have different security configurations. In a domain, caches equipped in the forwarding devices, servers, and clouds have capabilities for data caching and storage. P1, P2, and P3 represent IoT data publishers in Domaina, Domainb, and Domainc, respectively. U1 and U2 denote IoT data users in Domaina; U4 and U5 represent IoT data users in Domainc; U3 represents a mobile User who moves from Location L1, Domaina to Location L4, Domainc, passing through Location L2 and L3, Domainb.

IoT data, such as transportation data or healthcare data, can be published by Publishers, such as mobiles, sensors, actuators, and RFIDs. They are distributedly cached or stored throughout network. In Fig. 1, IoT data published by P1 are cached at access points, routers, and BSs, and stored in the cloud and servers, which is depicted with the yellow circle. It is assumed that the nearby cache or storage points for the targeted data of the Users is the cloud at Domaina, BS and access point at Domainb, and router at Domainc. The Users can retrieve data from these cache or storage points.

In this scenario, IoT data publications and retrievals are decoupled from time and location. Besides the Publishers and Users, there are intermediate actors involved in data dissemination and caching, such as routers, storages, and BSs.

The IoT system shall support event-based and periodical IoT streaming data sharing among devices as well. As the typical IoT scenario, we consider the following transportation data sharing. The car sensor detects an event that the road segment X, street Y is frozen and slippery at 9 am on Dec. 11, 2016. It wants to provide this data to drivers, and as a further restriction, only to the drivers who, given their current location, are expected to reach Street Y in 10 minutes, or to residents of buildings with more than 50 people along road segment X. These data are sent to the network and distributedly cached, and the drivers on the street or the people living in the building retrieve these data. Herein, we utilize Data Manifest (DM) to manage the version of data, where only the information on the latest data chunks is included and the old data chunks are deprecated from the network according to the caching strategy. If the data are updated, a new DM is issued.

The access policy can be used to define the access conditions under which a User can access the IoT data in the aforementioned example scenario. It is described with a policy tree. PT. PT includes non-leaf nodes and leaf nodes. Each leaf node is associated with one attribute (e.g., building, location, road segment), while each non-leaf node is associated with a Boolean function derived from the access policy. In the example use scenario, the IoT data are restricted to the Users satisfying the policy as \{\("Driver s\) V \("ex pected to reach S treet Y in ten minutes\)\} W \{\("buildings\) V \("along road segment X\) V \("with more than 50 peole in\)\}, where V means restriction, only to the drivers who, given their current location, are expected to reach Street Y in 10 minutes, or to residents of buildings with more than 50 people along road segment X. These data are sent to the network and distributedly cached, and the drivers on the street or the people living in the building retrieve these data. Herein, we utilize Data Manifest (DM) to manage the version of data, where only the information on the latest data chunks is included and the old data chunks are deprecated from the network according to the caching strategy. If the data are updated, a new DM is issued.

The access policy can be used to define the access conditions under which a User can access the IoT data in the aforementioned example scenario. It is described with a policy tree. PT. PT includes non-leaf nodes and leaf nodes. Each leaf node is associated with one attribute (e.g., building, location, road segment), while each non-leaf node is associated with a Boolean function derived from the access policy. In the example use scenario, the IoT data are restricted to the Users satisfying the policy as \{\("Driver s\) V \("ex pected to reach S treet Y in ten minutes\)\} W \{\("buildings\) V \("along road segment X\) V \("with more than 50 peole in\)\}, where V means restriction, only to the drivers who, given their current location, are expected to reach Street Y in 10 minutes, or to residents of buildings with more than 50 people along road segment X. These data are sent to the network and distributedly cached, and the drivers on the street or the people living in the building retrieve these data. Herein, we utilize Data Manifest (DM) to manage the version of data, where only the information on the latest data chunks is included and the old data chunks are deprecated from the network according to the caching strategy. If the data are updated, a new DM is issued.

The access policy can be used to define the access conditions under which a User can access the IoT data in the aforementioned example scenario. It is described with a policy tree. PT. PT includes non-leaf nodes and leaf nodes. Each leaf node is associated with one attribute (e.g., building, location, road segment), while each non-leaf node is associated with a Boolean function derived from the access policy. In the example use scenario, the IoT data are restricted to the Users satisfying the policy as \{\("Driver s\) V \("ex pected to reach S treet Y in ten minutes\)\} W \{\("buildings\) V \("along road segment X\) V \("with more than 50 peole in\)\}, where V means restriction, only to the drivers who, given their current location, are expected to reach Street Y in 10 minutes, or to residents of buildings with more than 50 people along road segment X. These data are sent to the network and distributedly cached, and the drivers on the street or the people living in the building retrieve these data. Herein, we utilize Data Manifest (DM) to manage the version of data, where only the information on the latest data chunks is included and the old data chunks are deprecated from the network according to the caching strategy. If the data are updated, a new DM is issued.

The access policy can be used to define the access conditions under which a User can access the IoT data in the aforementioned example scenario. It is described with a policy tree. PT. PT includes non-leaf nodes and leaf nodes. Each leaf node is associated with one attribute (e.g., building, location, road segment), while each non-leaf node is associated with a Boolean function derived from the access policy. In the example use scenario, the IoT data are restricted to the Users satisfying the policy as \{\("Driver s\) V \("ex pected to reach S treet Y in ten minutes\)\} W \{\("buildings\) V \("along road segment X\) V \("with more than 50 peole in\)\}, where V means restriction, only to the drivers who, given their current location, are expected to reach Street Y in 10 minutes, or to residents of buildings with more than 50 people along road segment X. These data are sent to the network and distributedly cached, and the drivers on the street or the people living in the building retrieve these data. Herein, we utilize Data Manifest (DM) to manage the version of data, where only the information on the latest data chunks is included and the old data chunks are deprecated from the network according to the caching strategy. If the data are updated, a new DM is issued.
time, which makes the problem more complicated. To overcome such ossification brought out by end-to-end communications, flexible and fine-grained secure data dissemination for dynamic group of users can be achieved by enforcing a PT to each piece of data.

III. THREATS AND SECURITY REQUIREMENTS

In this section, we detail the threats and security require-ments for a typical IoT use scenario. Threats should be inhibited from an architectural level and IoT data should be only accessible for a specific set of Users irrespective of where it is cached. The threats often occur when a Publisher publishes IoT data, or an intermediate node caches data, or a User retrieves data. They can be mainly classified as impersonation attacks and man-in-the-middle attacks (MIMAs).

An impersonation attack can be defined as using an impersonated identity for malicious/selfish purposes. In the attack A4 in Fig. 1, the attacker impersonates P3 to publish the data. In attack A1 in Fig. 1, the attacker impersonates U1 to retrieve data from the cloud. For MIMA, the data, which are published by a Publisher and cached in routers, access points, BSs, or servers, can be retrieved from the network, and illegally modified by the attackers during transmissions or at caches. Attack A5 is to illegally modify the data when they are retrieved from P2 by U3; attack A3 is to illegally modify the data when they are cached at a BS at Domainb; attack A2 is to illegally modify the data when they are retrieved from the network by U2.

Besides these attacks, the Publishers need to restrict the capability of Users to retrieve the data they publish. As in the scenario in Fig. 1, the IoT data published by P1 are restricted with an access policy as: (U1 W U2 W U3) V (Not L3) V (Not U4). Thus, when U3 moves to the location L3, he/she cannot successfully retrieve the data published by P1 (as F1 in Fig. 1), and U4 cannot obtain the data from the network (as F2 in Fig. 1).

To inhibit these attacks and provide flexible fine-grained access control on IoT data, we identify the security require-ments as follows.

- S1. Integrity for IoT Data Name and Data: To guar-anteGateway IoT data name and IoT data to be unable to be illegally modified or replaced by the intermediate attackers, which might locate at routers, gateways, BSs, or access points. It approves the correct linkage between data name and data. It is to inhibit MIMA.

- S2. Efficient and Flexible Authorization: To enable Publishers to publish data with a publisher-defined policy on the fly, and enable the authorization to Users for accessing a flexible set of IoT data. Flexibility here means changeability and adaptability. That is, a different set of IoT data are required and authorized for a User to access for different scenarios. The access policy for the published IoT data is enforced by a Publisher when it is generated and is determined depending upon the demand from the situation. We can say that the authorizations are flexible depending on the context. It is to efficiently provide access rights for flexible sets of Users to access the data in the network.

- S3. Publisher Identity Authentication: To assure that Publisher can be proved to be the one as claimed. It is to inhibit impersonation attack targeting at Publishers.

- S4. User Identity Authentication: To guarantee that Us-er is the one as claimed. It is to inhibit impersonation attack targeting at Users.

For authentication services of S3 and S4, NOAs manage Publishers’ and Users’ identities and provide an authentica-tion services to all the entities in the network, which can be realized by AAA. Meanwhile, each data chunk is appended with a signature of its issuer to naturally authenticate the identity. Thus, the authentication services of S3 and S4 are assumed to be provided by default.

IoT data will be disseminated across multiple adminis-trative boundaries and can be used for multiple purposes. It could be used for, at the time of publishing, unknown purposes and the access policy can be formed on the fly depending on real-time demands. Furthermore, for a flex-ible IoT data sharing, data are published from Publishers, cached in the network, and retrieved only by a specific set of Users with diverse attributes in a secure way. It is hence necessary to efficiently and inherently prevent unauthorized access, MIMA, and illegal publication and retrieval.

IV. PROPOSED DPD-ICIOT SCHEME

A. DPD-ICIOT Overview and Notations

To meet the security requirements described in Section III, we incorporate CP-ABE [21] and propose the DPD-ICIOT scheme in order to provide flexible access control while inhibiting the impersonation attacks and MIMA. A CP-ABE based scheme can provide fine-grained access control in a distributed manner. With it, each User is associated with a set of attributes based on which the User’s private key(s) is generated. When a Publisher encrypts each piece of
data, M, he/she specifies an access policy which is
expressed in terms of PT as described in Section II.
M is encrypted under the PT . An example of PT will
also be given later in Fig. 3 (a). CP-ABE usually
consists of four algorithms: Setup, Encryption(PK,
M, PT ), KeyGen(M K, S ), and Decrypt(CT , PriK).
These mathematical functions are detailed in the
Appendix.

CP-ABE can greatly improve the efficiency and
expe-rience of secure data dissemination among
flexible and unpredictable group of users. First, as
described previously, it naturally embeds IoT data
access policies into data encryption through the
Publisher by specifying the access policies over
attributes. Only Users whose attributes match the
access policy are able to decrypt the data without
being concerned about where the data are cached.
Therefore, it is characterized by the flexibility on
access policy enforce-ment and self-included security
in the data. Second, CP-ABE is especially suited for a
distributed caching environ-ment which decouples
IoT data publication and retrieval. Through it, IoT
data can be used for, at the time of publishing,
unknown purposes and the intended Users cannot be
predefined through a group establishment procedure
in ICIoT. Third, using CP-ABE, Publishers define the
access policy for restricting user access and then the
encrypted data can be stored anywhere on the
network without worrying about the unauthorized
access.

In DPD-ICIoT, a trusted third party, DSA, is
introduced to provide mediation services between
Publishers and User-s. DSA acts as a key server as
the common CP-ABE based scheme to provide the
data access rights to Users. Besides, it can also
provide attribute extraction services to Publishers.
That is, DSA assigns the start value, the start time,
update interval, and Hash functions for the attributes.
Then, it generates AMs including the attribute name
and the above information, and disseminates them in
the network for the Publishers to retrieve for data
encryption. NOA is another entity to provide identity
authentication services for Publishers and Users.
DSA together with NOA utilizes the ICN approach to
provide efficient, flexible fine-grained data-centric
access control and security services to establish trust
among IoT data, Publishers, and Users.

CCN is a typical ICN network architecture. In
CCN, Manifest has been very recently introduced to
describe a collection of Content Objects that
constitute one logical entity [26][27]. A Manifest is a
Content Object with a well-known payload format
and a Data-Type of “Mani fest”, rather than a normal
Content Type of “Data”. Content Object here
represents one data chunk and logical entity denotes a
set of chunks, which form one piece of data. In DPD-
ICIoT, to employ the merits of CCN, AMs and DMs
are novelly introduced to describe the attributes and
data access policies. AMs are issued only by the DSA
and cached at the network for fast retrievals. It well
balances the centralized management and distributed
retrievals of attributes. DM also manages the version
of data. In the existing CP-ABE scheme, attributes
are stored and only retrievable from centralized
servers. In contrast, the at-tributes and access policies
are described in AM and DM, respectively, in DPD-
ICIoT. AM and DM are provided in an information-
centric way and cached in a distributed manner in the
network for fast retrievals.

Publishers register their IoT data’s attributes with
the D-SA and DSA issues the corresponding AMs to
the network. AMs are cached in the network using the
ICN approach. Users acquire permission to access a
flexible set of IoT data from the DSA based on the
attributes they hold.

When publishing IoT data, Publishers retrieve the
related AMs from the close copyholders in the
network and enforce the security policy to the data
based on these attributes. To provide efficient IoT
data sharing, we do not intend to encrypt the data
directly using CP-ABE, because of computing cost.
We employ a key chain mechanism to provide
efficient encryption.

For the key chain mechanism, access policy is not
directly enforced over data for data sharing among
dynamic groups of users. Instead, it is used to encrypt
K E K, which is a file-lockbox key [31]. It enables K
K to be shared among dynamic groups of users.
That is, as per a key chain mechanism, the IoT data
are encrypted with a symmetric key (S K) by a
Publisher using symmetric encryption algorithms,
such as AES. This S K is encrypted by a K E K to
protect the encryption key, S K, and then this K E K
is encrypted by the expressive policy based on these
attributes using CP-ABE, such that only the Users
with the intended attributes are able to decrypt it.
Finally, the Publisher publishes the encrypted data
and meanwhile he/she publishes the DM, in which
the access policy to encrypt the K E K is specified.

When one User intends to acquire a piece of IoT
data, he/she queries DSA to generate key(s) based on
its attributes. DSA provides the private key (PriK)
based on the User’s attributes to the User to authorize
him/her with access privilege. Sometimes DSA needs
to provide a set of PriKs to a User for accessing
consecutive IoT streaming data with automatically
updating attributes. In such a case, the User obtains
the latest DM from the network. He/she can use PriK
to decrypt and obtain the K E K and further S K for
the data if his/her attributes satisfy the access policy
specified in the DM. Finally, he/she uses this SK to decrypt the messages and obtain the original data.

In DPD-ICIoT, the building block of flexible data access authorization is introduced to provide the basic functions including key generation, encryption and decryption algorithms, and the key exchange and operations. Because of the dissemination of AMs in the network using the ICN approach, attribute updates may bring out frequent AM flooding, which is a challenging problem for ICIoT. To solve this problem, we further propose the AASM to enable automatic self-updates of attributes. It supports the access privilege of Users for a period within which the attributes are updated many times without querying DSA(s).

The proposed DPD-ICIoT scheme (whose security notations are listed in Table I) can greatly reduce inter-action times between Publishers and Users. Without this DPD-ICIoT scheme, all Users have to search the targeted Publisher by themselves without enough information and interact with this Publisher for data retrieval each time. Meanwhile, the proposed DPD-ICIoT scheme introduces

<table>
<thead>
<tr>
<th>TABLE I: Security Notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbols</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>PT</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>MK</td>
</tr>
<tr>
<td>PK</td>
</tr>
<tr>
<td>PriK</td>
</tr>
<tr>
<td>PriK_{s, A}</td>
</tr>
<tr>
<td>SK</td>
</tr>
<tr>
<td>KEK</td>
</tr>
<tr>
<td>MEK</td>
</tr>
<tr>
<td>KeyGen(MK, S)</td>
</tr>
<tr>
<td>Decrypt(CT, PriK)</td>
</tr>
<tr>
<td>MEnc</td>
</tr>
<tr>
<td>H(M)</td>
</tr>
</tbody>
</table>

the cached AMs in the network, which enables the retrieval of them from close copyholders without querying central-ized servers. The key chain mechanism is also employed for encryption efficiency. Further, the mechanisms to cor-respond mobility and automatic updating of attributes are also integrated.

B. Building Block: Flexible Data Access Authorization

To meet the requirements described in Section III, we propose a building block for the DPD-ICIoT scheme, through which Publishers can provide flexible access privi-lege for specific data, and flexible authorization from DSAs to Users can be realized. In this building block, CP-ABE is used for key generation for Users. It allows for flexible access control based on attributes. The mathematical foundation of CP-ABE is the arithmetic of cryptographic pairing [21]. Previous research showed practicality, feasibility, and usefulness of using pairing-based mechanisms to solve security problems in sensor networks, and a fast and lightweight pairing-based cryptographic library for sensor networks has been developed [28]. Pairing and attribute-based cryptography is also utilized for constrained devices [30][33][35].

We elaborate on flexible data access authorization with regard to four components.

1) Setup: DSA selects the bilinear group and master key, MK, and public key, PK. DSA provides PK to Publishers and Users. Meanwhile, it generates AMs and DMs accord-ing to the needs of Publishers, and then disseminates them in the network using the ICN approach.

2) Key generation: DSA generates the private keys for Users using function KeyGen(MK, S) based on the Users’ attribute set, S, and the updated attribute values according to AASM described in Section IV.C. It uses different random numbers when generating private keys for different Users. Thus, the Users hold different private keys even if they hold a completely identical set of attributes. After generation of these private keys, the DSA sends these private keys to Users. The Users then install these private keys with the corresponding attributes.

3) Encryption: After searching for a set of IoT data, the Publisher retrieves the related AMs using the Interest/Data paradigm from the network and obtains the current value for attributes using AASM. The Publisher forms a policy tree, PT, based on the attribute values according to its demands. Then, the Publisher encrypts message, M, with the key chain mechanism, where S K is used to encrypt data through symmetric encryption, KEK is used to lock S K, and the encryption in CP-ABE is used to encrypt KEK and MK, PK PriK PK PriK obtain {KEK/(PK,PT )} as described in Section IV.A.

If Users are compromised by the attackers, they should be excluded from the policy tree when encrypting data. We let DSA disseminate the revoked User list for revocation in the network. When Users
are reported to be compromised, their special attributes, such as IDs, form a list and are announced in the network through the ICN approach. Here, the Bloom Filter [41] can be used to summarize the revoked list for efficient sharing. When the Publisher wants to encrypt data, he/she retrieves the revoked list from the network and excludes these Users from data access by excluding their attributes from the access policy trees.

4) Decryption: The User selects the relevant private key from the PriK set as per the current time for the updated attributes. Then, the User decrypts the \( \{K\text{ E }K\}(PK, PT) \) using this private key using function Decrypt(CT, PriK).

The key exchange and operations in DPD-ICIoT are provided as in Fig. 2 (a) and (b), respectively. As in Fig. 2 (a), DSA holds the master key, MK, and public key, PK, after setup. Then, the DSA announces the PK in the network and Publishers Ps and Users Us can obtain it. The DSA generates the values for the attributes, and introduces AM to describe the attributes as later detailed in Section IV.C. The AM is disseminated in the network by DSA using the ICN approach. Apparently, keys will not bring out much overhead for DSA, but the cost for attribute storage increases with the number of attributes increasing.

For Users Us, DSA generates private keys (PriKs) corresponding to their attributes. The Users and DSA authenticate each other based on the authentication service provided by NOA. Then, they establish a shared key among them through key negotiation protocol, such as Diffie-Hellman key exchange. The shared key is used to securely distribute the PriKs to Users. After the provisions of PriKs, the User holds one or a set of PriKs that reflects its attributes. One PriK is issued for one authorized period as introduced in Section IV.C. Thus, the number of PriKs held by a User increases as the number of authorized periods increase. On the other hand, the memory occupancy for key storage is much lower for Users and Publishers, and it is independent of the number of attributes.

Fig. 2 (b) shows the IoT data encryption, retrieval, and decryption operations in DPD-ICIoT. First, Publisher P considers the policy for encrypting the data and the attributes included in the policy. Then, Publisher P retrieves the desired attributes from the close caches. That is, he/she

issues the Interest for an attribute with the name as the attribute name to the network (Step 1 in Fig. 2 (b)). Besides, the AM Interest packet is appended with Publisher P’s signature, SigP, for P’s identity verifications. Then, the caches opportunistically with the AM(s) verifies SigP (Step 2 in Fig. 2 (b)) and replies with the desired AM(s) if the verification passes (Step 3 in Fig. 2 (b)). With the AASM, Publisher calculates the attribute values for the different authorized periods. Publisher selects one SK to encrypt the data, locks the SK with K E K, and further encrypts K E K with policy tree, PT, through CP-ABE with the attribute values in these authorized periods (Step 4 in Fig. 2 (b)).

For data retrieval, User U issues Interest for data to the network (Step 5 in Fig. 2 (b)). This Interest packet is also appended with User U’s signature, SigU . Either the Publisher or caches reply with the relevant DM and the encrypted data with Publisher P’s signature (Step 6 in Fig. 2 (b)). Here, DM has the format as the DM in Fig. 4. The dotted lines for Steps 5 and 6 in Fig. 2 (b) imply that the data may be retrieved from the intermediate caches or Publisher. It should be noticed here that the DM and the encrypted data are retrieved sequentially. DM is retrieved by issuing the Interest with data name. After receiving the DM, the data chunk sequence numbers (SeqNums) can be obtained from the payload of the DM. Then, User U requests the encrypted data using the data name with these sequence numbers. After receiving the encrypted data, User U verifies their authenticity by verifying the Publisher signature. Then, User U decrypts \( \{K\text{ E }K\}(PK, PT) \) in DM using CP-ABE decryption with the relevant PriK in the authorized period as in AASM. If PriK can satisfy the policy tree described in DM, SK can be
successfully obtained by User U. Otherwise, it cannot. Then, User U utilizes this SK to decrypt \( \{M\}SK \) and obtains the original data (Step 7 in Fig. 2 (b)).

Through the above setup and operations, the Publisher can specify a group of IoT data by encrypting a message using the key chain mechanism. Only the Users that hold the attributes satisfying a specific policy tree can decrypt the corresponding KEK, further SK, and finally the IoT data. Thus, it solves the problem of authorizing a User to access the selected set of IoT data.

All these AMs and DMs are issued with expiration time. When the expiration time is reached, the manifest will automatically expire. The periodical attribute update is a challenging issue in the building block when using the ICN approach, because the AMs including newly updated attributes should be input into the whole network to replace the old one.

Consider the example on transportation described in Section II to introduce the attribute update problem. There are five attributes in this scenario, “Driver”, “T imeoreachstreetY”, “Building”, “PeopleNum”, and “RoadSegX”. The policy tree for that use scenario is shown in Fig. 3 (a). Fig. 3 (b) shows the attribute value updates for 5 attributes in the example scenario. It is assumed that these 5 attributes are issued and updated simultaneously. With time flies, 5 attributes are updated 5 times as in Fig. 3 (b). It means that 5 new AMs for each of these attributes need to be disseminated in the network to replace the old ones. As in the simple network in Fig. 3 (c), even for the case with only 5 attributes, new updated AMs should be disseminated in the whole network for a total of 25 times, which brings out much traffic overhead. Hence, this procedure can be strenuous on network resources; therefore, besides the provision of the basic function to provide flexible access control and prevention of MIMA and impersonation attack, we further design AASM for efficient attribute update as described in the next subsection.

C. AM, DM and AASM

Here, we elaborate the details of AM, DM and the update mechanism for AM, AASM. AM is the manifest that describes the attribute value. AM is generated by DSA and disseminated over the network. It can be retrieved by the Publishers for encryption with CP-ABE. When a Publisher wants to publish data with a policy tree, PT, he/she first retrieves the corresponding AMs from the network using the Interest/Data paradigm in CCN, where an Interest with attribute name is issued and data with related attribute value information is returned.

The format of AM is depicted as in Fig. 4. The AM has the field of attribute name, such as DS A1/ServiceX/AttributeY. For the content of AM, the attribute start value, Start Time, Update Interval, Hash Function for update, and Hash(AM) are included. Hash(AM) is the hash value of the whole AM, which is used to assure the integrity of the data and linkage between the attribute name and the content. Other parameters except H(AM) are used for the provision of attribute value and to automatically update it.

![Fig. 3: Attributes updates for example use scenario](image1)

![Fig. 4: Attribute manifest and data manifest](image2)
The format of attribute name is defined as “DS A I D/S ervic T y pe/Attr ibute I denti fier”. DSA is to specify the identifier of DSA who generates the attribute and provides the AM. It is to separate the services from different DSAs. Service Type denotes the category of the attributes, which could be transport, environment monitoring, healthcare, etc. Attribute identifier shows the concrete attribute belonging to a specific category, which can also be a hierarchical structure depending on the needs. In the example in Fig. 4, the attribute name in the AM is DS A1/S erviceX/AttributeY. Further, consider the example on transportation described in Section II. If the DSA for Publishers in the use case is DSA2, the attribute names included in the policy tree can be DSA2/Transport/Driver, DSA2/Transport/TimeToReachStreetY, DSA2/City/Building, DSA2/City/PeopleNum, and DSA2/City/RoadSegX. Based on the attribute names, Publishers can easily retrieve the attributes using the names from the network.

DM is used to provide data feature descriptions including data access policy, PT , as in Fig. 4. In the payload of a DM, it includes the fields of access policy tree (PT ), (K E K){PK, PT }, data chunk sequence numbers, the hash value of this DM, publication time, version, and other features, such as hash value for each data chunk. Among them, data chunk sequence numbers are used to form the names of data chunks as data name appended with “/SeqNum”, which are used to retrieve data chunks after acquiring the DM. Publication time and version is for management of data, the period to afford the capability to read data, the period to afford the capability to read data. When a User intends to retrieve data, he/she first obtains the latest DM and obtains PT , (K E K){PK, PT }, and H(DM). He/she verifies H(DM), which assures the integrity of DM and linkage between data name and the content. Then, he/she decrypts (K E K){PK, PT } with the PT and the corresponding PriK to obtain K E K, and uses K E K to decrypt (K E K)K E K to obtain the S K. He/she also obtains chunk sequence number for all subsequent chunks, and retrieves all the encrypted data chunks one by one. Finally, he/she utilizes S K to decrypt all these encrypted chunks and obtains the original data.

In the DPD-I CloT scheme, AMs are disseminated throughout the network using the CCN approach, which can be potentially cached at any router in the network. When attribute update occurs, all these cached AMs should be replaced with the latest and updated AMs. Obviously, there is no method to explicitly find the location of all these AMs and then replace them through current technologies in CCN, because there is no location identification in CCN. Even though these can be realized, there is a considerable cost involved in attribute updates, because the DSA needs to notify all these routers holding the cached AMs one by one as described in Section IV.B.

To solve this problem, we propose AASM to enable the attribute values to be automatically self-updated. In AASM, the attribute start value and the update method is recorded in AM to enable the Publisher to obtain the current attribute value. With this self-updated mechanism, the attribute is updated automatically to eliminate the difficulty and cost for attribute update in the network.

When the Publisher obtains the AM, he/she can simply obtain the current attribute value through the

\[
\text{CurrentV} = \text{Hash}(\cdot \cdot \cdot \text{Hash}(\text{HashF}(\text{S tartV})) (i)) \\
\text{CurrentT} = \text{S tartT} \text{e} \text{U pdateInterval}
\]

where CurrentV is the omission of Current Value for an attribute. CurrentT is the omission of Current T ime, S tartT is the omission of S tart T ime of an attribute, and HashF is the omission of HashFunction in AM. The floor function forCurrentT \text{S tartT} is used to calculate the total \text{U pdateInterval} update times. Each time, attribute will be updated as the hash value of the previous attribute value. For example, we assume that CurrentT is 12:00 on Dec. 1, 2016, the S tartT is 0:00 on Nov. 20, 2016, and the U pdateInterval is one day. Then, the update times is 11 and 11 times of the hash value for StartValue is the current attribute value.

When generating a key, a DSA not only needs to consider automatic attribute update, it also needs to consider IoT data type, event-based IoT data or consecutive IoT streaming data. For event-based IoT data, attribute values within limited period will be utilized. In particular, for consecutive IoT streaming data, the period to afford the capability to read data should be considered carefully, because the attributes may be updated many times during this period.

Take the attribute values in the example scenario in Fig. 3 for example. We assume that the driver publishes the traffic data with the same policy for restricting the data access. Users with attributes, such as driver, should be afforded the capability to read the streaming transport data. The attribute value for Driver is changing as time flies. If the attribute value at t0 is used for key generation for Driver, he/she will be unable to access the data when it reaches t1, t2, t3, t4, t5, because the attribute values used for encryption keep changing. Thus, when generating the key for
Users, DSA should provide a set of private keys to her, where each key is afforded with time period for usage.

In this example, when a driver requests for key generation with the authorized period from t1 to t5, DSA will calculate with function PriKt1, Driver = KeyGen(M K, S 1 ), PriKt2, Driver = KeyGen(M K, S 2 ), PriKt3, Driver = KeyGen(M K, S 3 ), PriKt4, Driver = KeyGen(M K, S 4 ) to allow the driver to read the IoT data in this period, where S 1 , S 2 , S 3 , and S 4 are the different attribute value set at t1, t2, t3, and t4, respectively. DSA provides key set as {(PriKt1, Driver , t1), (PriKt2, Driver , t2), (PriKt3, Driver , t3), (PriKt4, Driver , t4)} to her. For the decryption of data, when she retrieves the data published between t2 and t3, she can decrypt the data using PriKt2, Driver.

With the AASM, AMs do not need to be replaced from the caches when attribute updates happen, and Users do not need to request DSA to generate new keys when updating the attributes in the DPD-ICIoT scheme. In contrast with the existing mechanism, the flooding of AMs for update as in Fig. 3 (c) becomes unnecessary.

**V. SECURITY ANALYSIS AND CHARACTERISTICS**

In the DPD-ICIoT scheme, the ICN approach is utilized to provide AMs and DMs; key chain mechanism is used for cryptographic operation efficiency; each User may be provided with several private keys for a corresponding DSA; and automatic attribute updates are performed as in Sections IV.B and IV.C. The hash function is used to generate the current values of attributes, which does not violate or change the mathematical foundation of CP-ABE and remains pairing and secret sharing. The proposed DPD-ICIoT scheme is therefore as secure as CP-ABE.

Besides preserving the security, the proposed DPD-ICIoT scheme offers the following properties, which meet the security requirements, S1 to S4, described in Section II.

- Integrity of IoT data name and data. We include the hash value of AM and DM into the manifests to assure the integrity of them. Meanwhile, the hash values for data chunks can be inserted into the DMs to check their integrity. If the immediate attackers modify AM, DM, or data chunks, this MIMA can be easily discovered, because the hash value of the data name and data will not equal the value calculated with the hash function. It enables the proposed DPD-ICIoT to satisfy the security requirement S1.

- Symmetric key establishment: The Publisher publishes the data encrypted by symmetric key for efficiency. A User can know the S K after the decryption of K E K(PK,PT ) and {S K}K E K in DM.

- Flexible authorization: The set of attributes corresponding to flexible group of Users can be included into the policy tree, PT , which is used in the encryption of S K. It enables the data to be readable only for a targeted set of Users, which may not be known beforehand.

- Attribute Self-Updates: The attributes are enabled to be automatically updated. The Publishers can efficiently get the current value for attributes and perform encryption. It together with symmetric key establishment and flexible authorization enables the DPD-ICIoT to satisfy the security requirement S2.

- Publisher and User identity authentication: The NOA can provide identity authentication services for these entities. It enables the DPD-ICIoT to satisfy the security requirements S3 and S4.

**VI. SYSTEM EVALUATIONS**

With the existing CP-ABE scheme, all the attribute values and attribute updates need to be provided through centralized servers, such as attribute server and DSAs. In contrast, the attribute values are described in AMs and retrieved from close caches. Herein, we perform system evaluations to compare the existing CP-ABE scheme with the proposed DPD-ICIoT scheme. We consider that the metric for comparison is the ratio between the bandwidth cost of the DPD-ICIoT scheme at the lowest performance situation with at most one cached copy in one domain and the bandwidth cost for CP-ABE. The bandwidth cost is defined as the bandwidth consumption for communications.

Assume that the network is divided into many domains. In each domain, one piece of AM or DM or data chunk can only be cached at most once, which is the lowest performance for CCN. If more AMs are cached, the cost for AM retrieval will be reduced further. CCN is utilized as the method for data retrieval.

Based on the above assumptions, a proposed network can be modeled as a undirected, connected graph G = (V ; E), where V is a finite set of vertices (network nodes), and E is the set of edges (network links) representing connection of those vertices. N denotes the total number of nodes in V . It is assumed
that each domain has the same size and K represents the number of nodes in one domain. For each domain, there are m gateway for connecting globally with other domains. Thus, the total number of gateways is m • N/K.

TABLE II: Notations for System Evaluations

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC^{AM}</td>
<td>Total bandwidth cost to retrieve AM through the DPD-ICIoT scheme</td>
</tr>
<tr>
<td>TC^{CSE}</td>
<td>Total bandwidth cost to retrieve AM through the existing CP-ABE scheme</td>
</tr>
<tr>
<td>TCR</td>
<td>The ratio between TC^{AM} and TC^{CSE}</td>
</tr>
<tr>
<td>BCLD</td>
<td>Bandwidth cost for local domain AM retrieval through the DPD-ICIoT scheme</td>
</tr>
<tr>
<td>BC^{LD}</td>
<td>Bandwidth cost for local domain AM retrieval through the existing CP-ABE scheme</td>
</tr>
<tr>
<td>BCGN</td>
<td>Bandwidth cost for inter-domain AM retrieval through the DPD-ICIoT scheme</td>
</tr>
<tr>
<td>BC^{GN}</td>
<td>Bandwidth cost for inter-domain AM retrieval through the existing CP-ABE scheme</td>
</tr>
<tr>
<td>N</td>
<td>Total number of nodes in the network</td>
</tr>
<tr>
<td>K</td>
<td>Average number of nodes in one domain</td>
</tr>
<tr>
<td>m</td>
<td>Average number of gateway in one domain</td>
</tr>
<tr>
<td>d</td>
<td>Average connection degree for one node</td>
</tr>
<tr>
<td>l</td>
<td>Average number of physical hops for one node to send packets to another node in one domain</td>
</tr>
<tr>
<td>L</td>
<td>Average number of physical hops to send packets from one domain to another domain</td>
</tr>
<tr>
<td>PS^{Type}</td>
<td>Packet size for type of packet</td>
</tr>
<tr>
<td>g</td>
<td>Total number of cached induplicate AMs</td>
</tr>
<tr>
<td>T</td>
<td>Average number of copies for one AM</td>
</tr>
<tr>
<td>R</td>
<td>Average number of update times for AM</td>
</tr>
<tr>
<td>Pt</td>
<td>The probability for intra-domain AM retrieval</td>
</tr>
<tr>
<td>t</td>
<td>Average retrieval times for one AM</td>
</tr>
<tr>
<td>a</td>
<td>Constant</td>
</tr>
</tbody>
</table>

Let l be the average physical hops for one node to send packet to another node in one domain, and L be the average number of hops to send packets from one domain to another domain. The packet size is assumed to be PS^{Type}. That is, Interest size is PS^{Type}, and AM packet size is S^{AM}. To transmit one packet in one domain, the bandwidth cost consumed for transmission is 1 • PS^{Type}.

It is assumed that g denotes the total number of cached induplicate attributes in the entire network. We assume each attribute is associated with one AM. Let f be the average number of copies for each AM. It is assumed that the attributes are averagely updated R times during the period that one Publisher uses it, and each data can only be cached at most once in one domain. We assume T to be the average retrieval times for one piece of AM in a period by different Publishers. Let Pl be the probability for intra-domain AM retrieval when a AM request occurs. The inter-domain AM retrieval occurs with the probability 1 − Pl. We assumed that AASM can be used throughout the period in the DPD-ICIoT scheme. That is, after AM is retrieved, Publishers do not need to retrieve it from the network again. The notations for performance analysis are summarized in Table II.

The objective is to model the bandwidth cost for AM retrievals in the proposed network in a period. The total cost consumed during a period equals to the sum of the cost consumed in AM retrieval procedures.

We need to model the bandwidth cost of AM retrievals during a period through the DPD-ICIoT scheme. For the AM retrieval, the Publishers obtain AM from the local domain with probability PL and from other domains with probability 1 − PL. Here, we do not consider the complex In contrast, we assume there is a centralized server, which is located at one domain in the network to provide AMs. We assume that the Publisher also sends out Interests Based on (2), (3), (4), the total bandwidth consumed for all AM retrievals during one period is further obtained as:

\[
BCGN = (PS^{Type} + S^{AM}) \cdot l + (PS^{Type} + S^{AM}) \cdot L + (PS^{Type} + S^{AM}) \cdot l
\]

For global inter-domain AM retrieval, we assume that the path for forwarding AM data packet is just the reverse path for forwarding AM Interest packet. The total band-width cost includes the cost consumption in local domain and foreign domain, and the cost for forwarding packets between the local domain and forwarding domain. Thus, we obtain the total cost consumed for one inter-domain AM retrieval procedure as follows.

where BCLD and BCGN denote the bandwidth cost for local domain AM retrieval and global area inter-domain AM retrieval, respectively, through the DPD-ICIoT scheme. We assume that the local domain network is a small-world network, which holds the property that the average number of physical hops l between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes in a network [24][25]. Thus, we assume l = a • log(K), where n is the number of nodes in the network. We also assume the network for connecting the gateways is also a small-world network. Thus, it can be assumed that L = a • log(m • N/K).

For intra-domain data retrieval, the consumer sends out the AM Interest packet and obtains the AM data packet with an average of 1 physical hops. Thus, the total bandwidth cost consumed for one intra-domain AM retrieval procedure is given as follows.
situation on caching, and just assume the data are pre-cached in f times in the whole network. It can be obtained that \( p_L = f / (N/K) \) in the DPD-ICIoT scheme. Then, we obtain the total bandwidth cost consumed in the AM retrieval procedures in DPD-ICIoT in a period as follows.

\[
T_{\text{CPD-ICIoT}} = \sum_{i=1}^{K} \left\{ p_L \cdot B_{\text{CLD}} + (1 - p_L) \cdot B_{\text{CGN}} \right\}
= \sum_{i=1}^{K} \left\{ \frac{f}{N/K} \cdot B_{\text{CLD}} + (1 - \frac{f}{N/K}) \cdot B_{\text{CGN}} \right\}
\]

where \( B_{\text{CLD}} \) and \( B_{\text{CGN}} \) denote the bandwidth cost for local domain AM retrieval and global area inter-domain AM retrieval, respectively, through the DPD-ICIoT scheme.

We assume that the local domain network is a small-world network, which holds the property that the average number of physical hops \( L \) between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes in a network \([24][25]\). Thus, we assume \( L = a \cdot \log(n) \).

For intra-domain data retrieval, the consumer sends out the AM Interest packet and obtains the AM data packet with an average of \( L \) physical hops. Thus, the total bandwidth cost consumed for one intra-domain AM retrieval procedure is given as follows:

\[
B_{\text{CLD}} = P_{\text{Interest}} \cdot 1 + P_{\text{AM}} \cdot 1
= a \cdot \log(K) \cdot (P_{\text{Interest}} + P_{\text{AM}})
\]

For global inter-domain AM retrieval, we assume that the path for forwarding AM data packet is just the reverse path for forwarding AM Interest packet. The total band-width cost includes the cost consumption in local domain and foreign domain, and the cost for forwarding packets between the local domain and forwarding domain. Thus, we obtain the total cost consumed for one inter-domain AM retrieval procedure as follows.

\[
B_{\text{CGN}} = (P_{\text{Interest}} + P_{\text{AM}}) \cdot 1 + (P_{\text{Interest}} + P_{\text{AM}}) \cdot L
= (P_{\text{Interest}} + P_{\text{AM}}) \cdot 1
\]

Based on (2), (3), (4), the total bandwidth consumed for all AM retrievals during one period is further obtained as:

\[
T_{\text{AM}} = \sum_{i=1}^{K} \left\{ \frac{f}{N/K} \cdot B_{\text{CLD}} + (1 - \frac{f}{N/K}) \cdot B_{\text{CGN}} \right\}
\]

In contrast, we assume there is a centralized server, which is located at one domain in the network to provide AMs. We assume that the Publisher also sends out Interests and the server replies with AMs. Because there is no cached AMs through this approach, \( p_L = 1/N/K \) for the existing CP-ABE scheme. Each time an attribute update happens, the Publisher needs to retrieve an updated AM from the server. The total bandwidth cost consumed by the existing CP-ABE scheme can be represented as follows.

\[
T_{\text{CP-ABE}} = \sum_{i=1}^{K} \left\{ P_{\text{Interest}} \cdot 1 + P_{\text{AM}} \cdot 1 \right\}
= \sum_{i=1}^{K} \left\{ a \cdot \log(K) \cdot (P_{\text{Interest}} + P_{\text{AM}}) \right\}
\]

The analytical model allows us to study bandwidth consumptions for different cases. To demonstrate the effectiveness of this model, first, we present typical performance results using the analytical model. We study the impact from the average retrieval times, \( R \), to the total bandwidth cost ratio (T CR), which is defined as the ratio of bandwidth cost for AM retrievals between the proposed DPD-ICIoT scheme and the existing CP-ABE, where CP-ABE is only utilized. That is, \( T\text{ CR} = T\text{ C}_{\text{CP-ABE}} / T\text{ C}_{\text{CPD-ICIoT}} \)

According to \([20]\), the total number of unique automatic system networks (ASNs) currently is around 5 • 104. Thus, for the purpose of the demonstration in our numerical examples, we assume that total number of nodes, \( N \), is set to be 106, and \( K \) is assumed to be 500, which is reasonable according to \([20]\). We assume that the Interest packet size, \( P_{\text{Interest}} \), is assumed to be 100 bytes, which is a small packet. The AM packet is is a little bit
larger than PS Interest, which is assumed to be 500 bytes.

The average number of gateways \( m = 10 \), the average number of copies for one piece of data for connection degree for the nodes in the domain \( d=3 \), \( g \) is assumed to be 108. We vary the \( R \) from 2 to 100 based on (5) and (7), and obtain the total bandwidth cost ratio, TCR, as Fig. 5.

From Fig. 5, we can see that the TCR drops considerably as the average update times increase. When the average update times reach 10, the bandwidth cost for attribute retrieval by the DPD-ICIoT scheme is reduced to be lower than 10% of the cost of the existing CP-ABE scheme. In the investigated situation, one AM is cached at most once in one domain. If more AMs are cached, the average number of hops for retrieving AMs is further reduced, and the TCR becomes much lower than the results in Fig. 5. Obviously, the DPD-ICIoT scheme can considerably reduce the bandwidth cost for attribute retrieval compared with the existing CP-ABE.

![Fig. 5: Ratio of total bandwidth cost](image)

**ACKNOWLEDGMENT**

This work is partially supported by the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Young Scientists (B) No.16K16054.

**REFERENCES**


