Enabling Attribute Based Encryption as an Internet Service

Runhua Xu* and James B.D. Joshi†
School of Information Sciences
University of Pittsburgh
Pittsburgh, USA
*runhua.xu@pitt.edu, †jjoshi@pitt.edu

***Abstract*—Internet enabled services and technologies are changing the way we use and manage massive amounts of data. More and more users and organizations are increasingly relying on cloud storage services for data management. In such Internet enabled environments, protecting sensitive data is increasingly becoming very crucial. Attribute-based Encryption (ABE) based approaches have been recognized as very promising for data protection in such environments. ABE approaches support data confidentiality and fine-grained access control in Internet-based environments, which include Internet of things (IoTs) and a plethora of heterogeneous mobile devices that enable large scale applications. However, in IoTs and mobile applications the limited computational resources and finite battery power of devices make it very difficult to use ABE schemes because of their heavy computational requirements. Although outsourced computational techniques have been applied in partial ABE schemes to address such issues, a unified platform that supports all aspects of data protection in an Internet-based open environment as well as fast cryptographic operations and decentralized authorities, etc., is still lacking. In this paper, we propose a novel Attribute Based Encryption as a Service (ABEaaS) that can be leveraged for data protection in the Internet environments. We propose an ABEaaS framework that can be easily deployed and present related security and performance analysis.**

***Index Terms*—Internet Service; Cloud Computing; Data Access Control; Attribute Based Encryption; Cloud Storage Service**

I. Introduction

Recently, cloud storage service has been gaining significant ground with potential "infinite" storage size, the convenience of synchronization, and the ease of access at any time and from anywhere. More and more users and organizations are thus increasingly utilizing cloud storage services to manage data. At the same time, with the increasing popularity and adoption of mobile devices (e.g., laptops, pads, cell phones, and IoT devices), cloud storage services are extending their support for applications on multiple-devices including mobile devices because of their varying resource constraints. However, there are significant challenges towards making application secured adaptable to multiple-device scenarios.

Although cloud storage services provide users with convenience, they introduce newer problems of security and privacy. A survey from Gartner [1] predicts that at the end of 2016, more than 50 percent of 1,000 global companies would have customer-sensitive data stored in the public cloud. However, there is also a significant insider threat of cloud storage providers themselves secretly analyzing or leaking customers' sensitive data. This raises a significant question about how users are able to fully trust the cloud storage providers with regards to protecting their sensitive data stored there. Towards protecting customers' sensitive data cryptographic techniques and access control mechanisms should be adopted. In particular, to tackle the above-mentioned challenges, Sahai and Waters [1] introduced the Attribute-based Encryption (ABE), which provides fine-grained accesses control and data security to outsourced data stored in the cloud.

ABE has been found to be a promising approach to enable cryptography based access control mechanisms and various extensions have been proposed in the literature [2]–[6]. However, ABE adoption in the IT industry is still a significant challenge. There are two key challenges towards its adoption. First, as a key part of an ABE based infrastructure, the authority center is very hard to deploy in practice. Because it is particularly difficult to deploy a global trust center for all Internet users. Second, though multi-device scenarios are pervasive, there is a limitation of computational resources and battery power needed to support encryption and decryption operations in small mobile devices, making ABE deployment in the mobile devices very difficult. Various research efforts have focused on employing outsourced computations to address resource constraints in mobile devices, as in [2], [6]–[12]. Currently, we believe there is a lack of an effective deployment approach to make ABE mechanisms available broadly as a service to support a broad set of mobile cloud applications that would benefit from the ABE schemes.

In this paper, we present an Attribute Based Encryption as a Service (ABEaaS) approach to address the second issue mentioned above. Using ABEaaS, we propose a mechanism to deploy ABE widely over various cloud platforms. The goal of our proposed work is, thus, to support the use of mobile cloud services with ABE for all types of resource-constrained mobile devices.

In Figure 1, we illustrate an application scenario where the proposed ABEaaS can be applied, where data is stored in the cloud storage under the protection of an ABE scheme that provides fine-grained access control. Furthermore, most of the computations in the encryption and decryption phases

[1]Gartner, Inc. is the world's leading information technology research and advisory company.

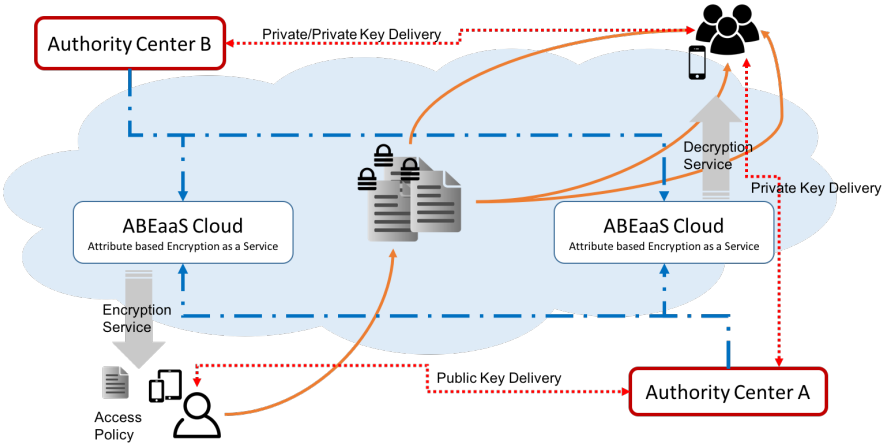


Fig. 1. Application scenario of ABEaaS

of ABE are packaged as a service provided by the cloud itself. The ABEaaS requires users' devices to support only a small part of cryptographic computations without any potential leakage of confidential computation. Suppose users want to store and share their data in the cloud storage environment. In data storing phase, they just need to pull some intermediate ciphertexts from ABEaaS cloud based on what authority center they named (*encryption service*). Then they encrypt the data by a simple computation with the specified access policy and the intermediate ciphertexts. In the data sharing phase, the data receivers first request the secret keys and temporal keys from the authority center. If their attributes satisfy the access policy, they can get the intermediate ciphertexts decrypted by the ABEaaS cloud with the temporal keys (*decryption service*). Finally, they recover the original data by using simple cryptographic computation with their secret keys.

The key contributions of the proposed work are as follows:

- We present a novel Attribute Based Encryption as a Service (ABEaaS) that can be used by applications in a mobile cloud environment. The goal of ABEaaS is to provide an ABE service and reduce the computational burden of users' resource-constrained mobile devices. Towards that the idea is to outsource most of non-confidential encryption/decryption operations of the ABE mechanism to the cloud.
- To enable ABEaaS we propose two parts: encryption service platform and decryption service platform, which are deployed in the cloud environment. To provide flexibility, the ABEaaS also has two different settings called *general setting* and *hybrid setting* to accommodate varying needs of the users.

The rest of the paper is organized as follows. In Section II, we present the concept of ABEaaS. Our proposed mechanism for ABEaaS is described in Section III. We discuss and analyze the security and performance in Section V and Section VI, respectively. Finally, we conclude the paper in Section VIII.

II. Attribute Based Encryption as a Service

A. Goal of ABEaaS

We propose an Attribute Based Encryption as a Service (ABEaaS) and the key goal is to construct a cryptographic service that can be used by mobile applications. The key requirements for such an ABEaaS service to be useful are as follows:

- *Security.* As ABE is essentially a cryptographic mechanism, the security of the service should be the first goal to be considered. Enabling it as a computing service means the security of this service module itself needs to be properly considered.
- *Elasticity and Scalability.* The ABEaaS implementation architecture should be elastic and scalable. That is, it should be easy to add/remove a node easily without impacting the service and it should scale well as the number of nodes grows.
- *Availability.* The architecture of the service should also be available all the time. One approach to that is through redundancy, which means the system has a backup for its service. It should not be affected when a single service point is down.

B. Overview of ABEaaS

1) ABEaaS: In this paper, the proposed *ABEaaS* refers to the secure delivery of attribute based encryption as a service via the cloud computing platform. Note that:

- *ABEaaS* should ensure the security assurance of the supported ABE schemes.
- *ABEaaS* should not introduce new security vulnerabilities into the system.
- *ABEaaS* a computing service based on different cloud platforms. Thus, it will inherit the advantages and disadvantages of cloud computing platform used.

An overview of the proposed ABEaaS is shown in Figure 2. Based on different cloud environments where encryption service is deployed, we have two settings for the ABEaaS:

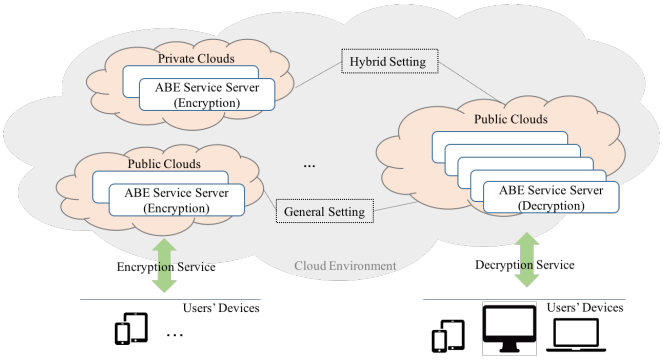


Fig. 2. Overview of ABEaaS

Hybrid Setting and *General Setting*. In the hybrid setting, the encryption and decryption service are deployed in private and public cloud, respectively. While in the general setting, both the encryption and decryption service are deployed on the public cloud.

The difference between hybrid setting and general setting is where the encryption service is deployed and what operations the encryption service include. The encryption service deployed in a private cloud can include more sensitive operations. From the users' perspective, compared to the general setting, the benefit of using the hybrid setting is that it can help users reduce one pairing computation and one exponential computation. Also the hybrid setting needs the support of private cloud infrastructure that is not generally available for common users. We prefer using general setting in the ABEaaS for the following reasons:

- Both these settings provide the users with significant performance, but the general can reduce deployment costs of the ABEaaS.
- In the data access scenario, the frequency of read access is much higher than the frequency of write access. Similarly, in the ABE scenario, the decryption operation has a higher frequency. The encryption service in the general setting would not be the bottleneck that affects the usage of the ABEaaS.

The hybrid setting could be the choice for the organizations which have private cloud infrastructure. In the following sections, we will focus on the general setting ABEaaS.

C. From ABE to ABEaaS

The ABEaaS is designed to be a unified platform which adapts for most of ABE schemes. Recent various ABE schemes have been proposed [1], [3]–[5], [13]–[16]. Because the difference in the specific ABE constructions, it is hard to make the ABEaaS available to every ABE scheme. We show how to choose proper ABE schemes that could be applied in our proposed ABEaaS framework.

1) Overview of ABE: Here, we briefly overview the ABE scheme. ABE scheme was first introduced by Sahai and Waters [1]. A typical *ABE* scheme consists of the following four probabilistic polynomial time (PPT) algorithms:

- *Setup*. The setup algorithm sets up the parameters for the ABE scheme, which includes public parameters and master secret parameters.
- *Key Generation*. The key-generation algorithm generates the necessary keys for the users.
- *Encryption*. Users use the encryption algorithm and specified access policy to encrypt the data.
- *Decryption*. Users use the decryption algorithm to decrypt the ciphertext data.

As per the *ABE* scheme, users only need to execute the encryption and decryption algorithms. Thus, the main services provided by the ABEaaS are for encryption and decryption.

2) CP-ABE and KP-ABE: The two major types of ABE schemes, Key-Policy Attribute Based Encryption (KP-ABE) [16] and Ciphertext-Policy Attribute Based Encryption (CP-ABE) [3], are designed for different application scenarios. The difference between KP-ABE and CP-ABE are the approaches to key generation and the types of attributes (attribute sets / attribute policy) that are used in encryption. However, the ABEaaS is not responsible for key-generation algorithm, and outsourced encryption computation does not include the computation that relates to the specific access attributes. Due to the similarity of the basic architecture of CP-ABE and KP-ABE, both KP-ABE and CP-ABE can be applied in our ABEaaS platform.

3) Access Structure: ABE has three different types of access structures: *AND-gate* structure, *Tree-based* structure, and *Linear Secret Sharing Schemes (LSSS)* structure. No matter what kind of access structures are used in the ABE schemes, the encryption components related to access structure is processed by users' local device. Thus our ABEaaS platform should be independent of the different types of access structures in the ABE schemes.

4) Unbounded ABE: Initial ABE schemes are set up with *bounded* attribute sets, i.e., the attributes used in ABE system should be defined in the setup phase. That indicates, if new attributes are introduced into the system, the ABE system should restart to make the new attributes available. Meanwhile, the previous ciphertext should be re-encrypted again if no proper attribute version management is provided. As a universal service for a number of users in the cloud environment, this *attributes-predefine* setting is unacceptable.

Thus the ABE schemes deployed in ABEaaS platform should be *unbounded*, which means the generated public parameters should not include components corresponding to the particular attribute sets in the setup phase. We believe that it is not a strict condition to apply the ABE schemes. By using the proper hash technique, most of the ABE systems can support unbounded attribute sets. In this paper, we adopt the work by Rouselakis and Waters [5] who present a typical large unbounded universe ABE scheme with new proof methods. Furthermore, this scheme will be used as an instance in the following section.

5) Technique to Outsource Computation: Inspired by the works in [10]–[12], the computation outsourcing technique is introduced into ABE schemes [2], [6]–[9]. The key idea is

to outsource partial computation to a powerful server without impact on the functionality and security of the ABE scheme.

Among these schemes, Hohenberger and Waters [2] propose a universal encryption outsourcing method that could be used to update all unbounded ABE schemes. For outsourcing decryption of ABE ciphertexts, Green et al. [6] give one universal solution. By applying these techniques, the ABEaaS can be a unified platform for all unbounded ABE schemes. Specifically, the ABEaaS will prepare a number of intermediate ciphertext components in advance for encryption. When users need to encrypt data, they just request these prepared intermediate ciphertext components from the ABEaaS platform and do simple computation under the specified access structures. In the decryption phase, users need to send the ciphertexts and temporary keys to the ABEaaS platform and receive the intermediate ciphertexts. Then the users need to do the final decryption computation and get the original data.

III. ABEaaS Architecture

In this section, we first show the overview of the ABEaaS architecture. Then we present each part.

A. Overview

The architecture of the ABEaaS is shown in Figure 3. The architecture has little difference for the hybrid and general settings, mainly, there is an additional component called the secret intermediate component in the hybrid setting. Further, encryption and decryption service could be independent of each other in the deployment setting.

B. Base Architecture

The base architecture of ABEaaS is based on a dual-master multi-slave framework. There are two manager nodes and several work nodes. Generally,each manager node is responsible for managing several work nodes. The work nodes provide the encryption or decryption services. As shown in Figure 3, manager nodes have the following core functions:

- *Entry and exit of the service*. It is responsible for handling the users' service request.
- *Management of work nodes*. It checks the status of the work nodes and manages them.
- *Management of service costs*. It manages the charge methods, such as charge by flow, charge by usage, and charge by time.

Note that there are two manage nodes in the ABEaaS platform. One is the main manager node, and the other one is a backup node to increase availability. The backup node synchronizes with the main node in real time And it will be activated by the cloud environment if the main manager node is down. Several work nodes are responsible for the specific computation of encryption/decryption service. As the computation task is a short-time job and does not include contextual computation, these nodes can work in parallel. That indicates each work node is responsible for a task independently. Once a service request is received, the manager node assigns available nodes with associated computation task. This basic architecture supports the goal of scalability and availability that we discussed in Section II-A.

C. Manager Node

The manager node in both encryption and decryption services includes following two functional modules:

- *Request Dispatcher (RD)* is responsible for handling the users' service requests. It receives the request and dispatches the request to an available work node. After receiving the computation result, the manager node sends it back to the user.
- *Work Node Management (WNM)*. It manages a number of work nodes in the cloud, supporting features like adding and removing work node.

D. Work Node

As outsourcing computation algorithm for encryption and decryption are different, it is hard to build a universal architecture for both of the services. Here we introduce the encryption service architecture and decryption service architecture, respectively.

1) Encryption Service: The encryption service node has the following components:

- *Authority Management*.
- *Pools Management*.
- *Random Selector*.
- *Attribute Intermediate Ciphertext Generator*.
- *Attribute Intermediate Ciphertext Pool*.

For the hybrid setting, there are two additional components, *Secret Intermediate Ciphertext Generator* and *Secret Intermediate Ciphertext Pool*.

Authority Management (AM) component is responsible for managing the public parameters (PK) from different authority centers. In the cloud computing environment, it is hard to require all users using a common authority center. Usually, the users of ABEaaS are registered to different authority centers. However, each intermediate ciphertext corresponds to a specific authority center's public parameters. That indicates the ABEaaS platform will use a number of public parameters from different authority centers. Thus it is necessary to have an authority management component to manage these public parameters.

Secret/Attribute Intermediate Ciphertext Generator (SICG/AICG) are the core components in the encryption service node. The *AICG* is in charge of generating intermediate ciphertexts used for attributes related computation. Additionally, for the hybrid setting, the *SICG* is for generating the main security intermediate ciphertexts with a specific PK.

Pools Management (PM), *Secret Intermediate Ciphertext Pool (SICP)*, and *Attribute Intermediate Ciphertext Pool (AICP)* are the three pool related components. Due to several components need to be calculated in the encryption service, it will increase the processing time for users' requests. To increase the efficiency of the service, we propose the pool

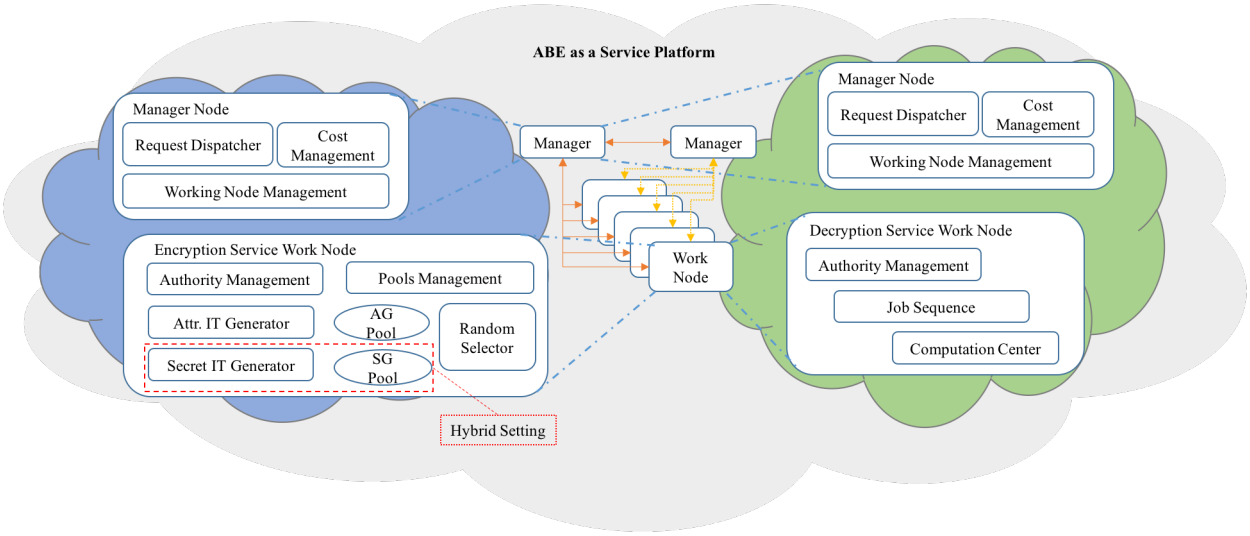


Fig. 3. Architecture of ABEaaS Platform

architecture to store the pre-calculated intermediate ciphertext. *SICP* and *AICP* are used to store the *Secret Intermediate Ciphertext (SIT)* and *Attribute Intermediate Ciphertext (AIT)* that are generated by SICG and AICG, respectively. *PM* component manages the two pools.

Even though the two pools (SICP/AICP) store a number of intermediate ciphertexts (SIT/AIT) pairs, each service request only needs one SIT tuple and one AIT tuple. It is necessary to have a *Random Selector (RS)* component to select a random SIT tuple and a random AIT tuple from the pools.

2) Decryption Service: The decryption service has three components: *Authority Management*, *Job Sequence*, and *Computation Center*. Comparing to the architecture of encryption service nodes, it is quite simple in decryption service nodes. The decryption service nodes are responsible for computing intermediate ciphertexts according to the temporary keys from the users. However, the frequency of using decryption service is several times than the frequency of using encryption service. Thus the efficiency of decryption service is the main challenge in the architecture of decryption service. As the delegation decryption is a pipeline computation model, we design a *Job Sequence* component to store the jobs with proper scheduling strategy. Then the decryption jobs are executed in parallel by several executors from computation center. The authority management is the same to encryption service node.

E. Workflow of ABEaaS Platform

Initialization. Both encryption and decryption service need to register authorities based on the initialization settings, which include several default authorities. Then the encryption service initializes the *SICP* and *AICP* by the *SICG* and *AICG* based on the public parameters, respectively.

Encryption. When users want to encrypt data, they send an encryption request to the service platform with the authority identifier as the parameter. The manager node receives the request and dispatches it to a work node with the authority identifier. The work node first checks the authority module to find whether the authority identifier exists or not.

a. If the authority identifier exists, the random selector will choose the corresponding tuples from the *SICP/AICP* randomly and return to manger node. Additionally, the selected tuples will be removed from the two pools.
b. If the authority identifier does not exist, the work node first queries the public parameters from the authority center and registers to the platform. The work node prepares the intermediate ciphertexts, fills the pools, and then does as in (a).

Then the manager node returns the results to the users.

Decryption. The decryption workflow is similar to encryption workflow. The only difference is that the decrypted intermediate ciphertexts are not prepared in advance. When computation jobs come into the job sequence, several work nodes compute the results in parallel. Then the finished jobs are returned to the users by the manager node.

IV. ABEaaS Implementation

ABE could be divided into KP-ABE and CP-ABE. In this section, we use unified algorithms to present an ABEaaS instance formally.

A. ABE Prototype in ABEaaS

Here is the prototype model of ABE scheme used in the ABEaaS Instance, which is extended from [2] and [6]. The presented ABE instance has the following six algorithms:

- $Setup_{authority}(\lambda, U) \rightarrow (PK, MSK)$
- $KeyGen_{authority}(MSK, S) \rightarrow (TK, SK)$
- $Encrypt_{service}(PK) \rightarrow (IT)$
- $Encrypt_{user}(PK, IT, AC, data) \rightarrow (CT)$
- $Decrypt_{service}(TK, CT) \rightarrow (\widetilde{CT})$
- $Decrypt_{user}(\widetilde{CT}, SK) \rightarrow (data)$

Note that the subscript of each algorithm indicates the executor of the algorithm, where *authority* represents the trust authority

center, *service* means the ABEaaS platform, and *user* indicates the users of ABE service. The parameter AC in algorithm $Encrypt_{user}$ represents access structure and attribute sets in CP-ABE and KP-ABE, respectively.

B. Initialization

1) Authority Initialization: Suppose that we have m authority centers all over the world and each authority center has its public parameters (public key). Let Au_x denote the *Authority Center* x, where $1 \leq x \leq m$. Then Au_x generates the PK_x and MSK_x as follows:

$$PK_x = (\mathbb{G}_x, p_x, g_x, h_x, u_x, v_x, w_x, e(g,g)^{\alpha_x}),$$
$$MSK_x = (\alpha_x),$$

where $\mathbb{G}_x$ is a random bilinear group with prime order p_x, $g_x, h_x, u_x, v_x, w_x \in_R \mathbb{G}_x$, and $\alpha_x \in_R \mathbb{Z}_{p_x}$. [2]

2) ABEaaS Platform Initialization: Suppose that we have a default authority setting list with size n. Let $L = (AuID_y, PK_y), 1 \leq y \leq n$ be the authority list, where $AuID_y$ is the identifier of authority center and PK_y is the public parameters that are pulled from the corresponding authority center. As shown in Algorithm 1, the algorithm first initialize storage structures. Then it requests all public parameters from the authority centers according to the default authority setting. Based on the public parameters of each authority, it generates the attributes intermediate ciphertexts and stores them into the pools.

C. Encryption Service Algorithm

As shown in Algorithm 2, the encryption service algorithm randomly selects attribute intermediate ciphertexts components from the pool. Then it returns the selected components and removes them from the pool. If the count of rest intermediate ciphertexts in the pool is not enough, the manager node will receive the $signal_{empty}$ and re-assign the request to another working node. Finally, the manager node marks the status of work node as $status_{empty}$.

D. Decryption Service Algorithm

The decryption service algorithm is shown in Algorithm 3. It first checks the id of the authority and finds the public parameters of the authority. Then it puts the tuple $(job_{id}, < pk_{id}, CT, TK >)$ into job sequence and keeps checking the that until getting the results.

E. Pool Maintenance

As described in *Encryption Service Algorithm* (Algorithm 2), it keeps using the intermediate ciphertext from the pool. Even though the size of the pool is much bigger than the requested attribute size, the pool will be exhausted sooner or later. The pool maintenance algorithm is designed for such case, which will be executed by an independent process. Due to its similarity to *Service Initialization Algorithm* (Algorithm 1), we do not give the specific algorithm description. The maintenance algorithm keeps checking the status of all work nodes. If the status of a work node is out of service by pool empty, it will fill the pool as required, which is similar to *line:8-22* in Algorithm 1. Then it marks the status of work node as $status_{working}$.

[2]Symbol "a $\in_R$ A" indicates a is randomly chosen from A.

Algorithm 1 Service Initialization with General Setting.

Input: $type_{op}$, the service type (encrypt/decrypt),
$type_{ABE}$, the ABE type (KP-ABE/CP-ABE),
$list$, the default authority setting list,
$size_{pool}$, the default size of pools.
Output: $m_{authority}$, a map for the authorities information,
m_{AICP}, a map for the AICP.

1: initialize the map, $m_{authority}$
2: **for** id in $list$ **do**
3: $pk_{id} \leftarrow$ request the public key from authority.
4: push $(id, pk_{id}) \rightarrow m_{authority}$
5: **end for**
6: **if** $type_{op}$ == "Encrypt" **then**
7: initialize the maps m_{AICP}.
8: **for** id in $list$ **do**
9: initialize a new list $list_{AICP}$
10: $pk_{id} \leftarrow m_{authority}[id]$
11: $s \leftarrow$ **random**($\mathbb{Z}_{p_{id}}$)
12: **for** $i = 0$ to $size$ **do**
13: **if** $type_{ABE}$ == "CP-ABE" **then**
14: $\lambda, x, t \leftarrow$ **random**($\mathbb{Z}_{p_{id}}$)
15: $C_1 = g_{id}^{\lambda} v_{id}^{t}, C_2 = (u_{id}^{x} h_{id})^{t}, C_3 = g_{id}^{t}$
16: add tuple $(\lambda, x, t, C_1, C_2, C_3) \rightarrow list_{AICP}$
17: **else**
18: $r, x \leftarrow$ **random**($\mathbb{Z}_{p_{id}}$)
19: $C_1 = w_{id}^{r}, C_2 = (u_{id}^{x} h_{id})^{r} w^{-s}$
20: add tuple $(r, x, s, C_1, C_2) \rightarrow list_{AICP}$
21: **end if**
22: **end for**
23: push $(id, list_{AICP}) \rightarrow m_{AICP}$
24: **end for**
25: **return** $m_{authority}, m_{AICP}$
26: **else**
27: **return** $m_{authority}$
28: **end if**

Note: the function **random(A)** generates random elements between 0 and $|A|$.

V. Security Analysis

The ABEaaS platform includes two major parts: the ABE scheme and the service architecture. We analyze the security requirements of ABEaaS in this section.

A. Assumption

Before the security analysis, we list two general assumptions first:

- The cloud provider is *honest-but-curious*, i.e., that the cloud provider is honest, but try to collect users' information. This is a general assumption in the literature.

Algorithm 2 Encryption Service.

Input: id, the authority id of the user,
$size_{attribute}$, the number of attributes size,
m_{AICP}, a map represented the AICP,
$m_{authority}$, the authorities information.
Output: $it_{attribute}$, the tuple of attribute intermediate ciphertext.

if id in $m_{authority}$ **then**
  pull $pk_{id} \leftarrow m_{authority}$
else
  execute the initialization with the id
end if
$list_{AICP,id} \leftarrow m_{AICP}[id]$
if $|list_{AICP,id}| > size_{attribute}$ **then**
  for $i = 0$ to $size_{attribute}$ **do**
    $index_{random} \leftarrow$ **random**$(|list_{AICP,id}|)$
    $it_{tuple} \leftarrow$ pop $list_{AICP,id}[index_{random}]$
    add $it_{tuple} \rightarrow it_{attribute}$
  end for
  return $it_{attribute}$
else
  return $signal_{empty}$
end if

Note: that $size_{att} << size_{pool}$, which indicates the size of requested attributes set is much smaller than the size of pool. $|A|$ denotes the size of list A.

Algorithm 3 Decryption Service.

Input: id, the authority id of the user,
S, the job sequences,
CT, the ciphertext,
TK, the temporary key of CP-ABE.
Output: $\widetilde{CT}$, the intermediate ciphertext.

if id in $m_{authority}$ **then**
  pull $pk_{id} \leftarrow m_{authority}$
else
  execute the initialization with the id
end if
push tuple $(job_{id}, < pk_{id}, CT, TK >) \rightarrow S$
for true **do**
  if status of job_{id} == $signal_{done}$ **then**
    $\widetilde{CT} \leftarrow S[job_{id}]$
    return $\widetilde{CT}$
  end if
  if time out **then**
    return $signal_{time.out}$
  end if
end for

- The communication between the service user and ABEaaS platform is protected using a secure channel. Thus the security issues about the data transmission will not be discussed.

B. Security of ABE

As one of the major parts, the ABEaaS applies the ABE schemes based on the principle specified in Section II-C, but does not change the secure base of the ABE schemes. We refer the readers to find the formal security proof from the works [2], [6] mentioned earlier.

C. Security of Service Architecture

In this section, we show the security analysis of another major part, service architecture, in the ABEaaS.

1) Security of Encryption Service Node: In the general setting, the sensitive modules are AICG and AICG pools. In the hybrid setting, there are two additional sensitive related modules, SICG and SICG pool. These modules are sensitive because they produce the intermediate components for the encryption.

The encryption service does not introduce new risk in security for the following reasons: i) The *AIC* is the intermediate information, but does not include any secret. ii) The *AIC/SIC* is disposable, i.e., when the intermediate ciphertext is used, it will be destroyed immediately. iii) The intermediate ciphertext is randomly selected from the pool. iv) Even though the *SIC* includes secret information, it is only used in the hybrid setting, but not in the general setting. Based on our assumption, the encryption service architecture satisfies the security requirement.

2) Security of Decryption Service Node: The decryption operation in the ABEaaS architecture is not complex. Essentially, decryption service is a delegation computation service. As shown in the security analysis of outsourced decryption in [6], the delegation computation does not introduce new security issues. As the decryption service architecture of the ABEaaS does not change the structure of delegation computation algorithm, we believe that our decryption service node does not introduce extra security issues.

Note that security issues related to the cloud environment, e.g., DDoS attack, in general can still be prevented.

VI. Performance Analysis

A. Scalability and Availability

The ABEaaS platform is based on the dual-master multi-slave architecture. The main manager node has a backup manager node with real-time synchronization. Backup nodes can be used to ensure a higher level of availability. Additionally, the availability of work node is also guaranteed by the design of multi work nodes. Due to the design of independent computing of each work node, it is easy to add/remove the work nodes without impacting the service computation. Thus the ABEaaS also provides scalability.

B. Efficiency of Using ABEaaS

1) Efficiency Estimates: We show the efficiency estimates from the users' perspective, as shown in Table I, theoretically. Compare to a specific ABE scheme [5], the ABEaaS scheme can help users to reduce all the bilinear map computation,

which is the major computation consumption in the pair-based cryptographic schemes. Further, the ABEaaS scheme also helps users reduce most of the exponentiation and multiplication computation.

2) Experiment Setting: A prototype system of the ABEaaS applying ABE scheme described in [5] is implemented by using a framework called Charm [17]. Charm is a framework for rapidly prototyping advanced crypto-systems based on Python. Our experiment is simulated on the MacBook Pro with 2.5 GHz Intel Core i7 processor and 16GB memory.

In the experiment, we design ten policy cases with the number of attributes from 1 to 10. For each policy case, we test user's encryption/decryption time using the original ABE scheme and the ABEaaS, respectively. Note that the recorded operation time is measured by the CPU execution time and the measurement unit is the millisecond.

3) Analysis of the Result: As shown in Figure 4, we compare the user's operation time between the original ABE scheme and the ABEaaS scheme in the General Setting. For user's encryption, the time of both schemes increases with the number of attributes. However, compared to the original ABE scheme, the ABEaaS scheme reduces the computation for users' device significantly, as shown in Figure 5(a). For the user's decryption service, the time of original ABE scheme increases with the number of attributes. While the time of ABEaaS scheme is flat, as shown in Figure 5(b). The experimental result is consistent with the theoretical analysis in Table I.

Based on the experiment result, we show further analysis on the performance of the ABEaaS system. As shown in Figure 5(a), the ABEaaS system could save user's computation time both in encryption and decryption. Compared with the traditional ABE user, the ABEaaS user will save more time when the complexity of access policy increases.

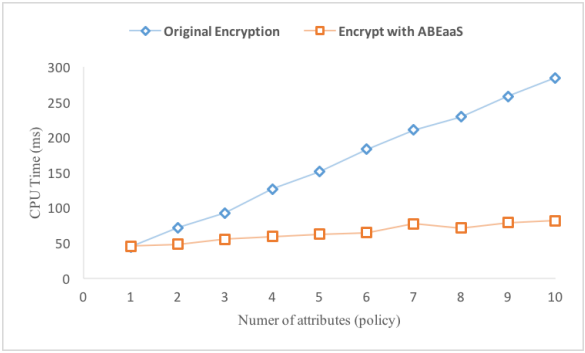
Here we define the advantage of using ABEaaS (P_{Adv}) of the ABEaaS system as:

$$P_{Adv} = \frac{time_{ABE} - time_{ABEaaS}}{time_{ABE}} \times 100(\%)$$

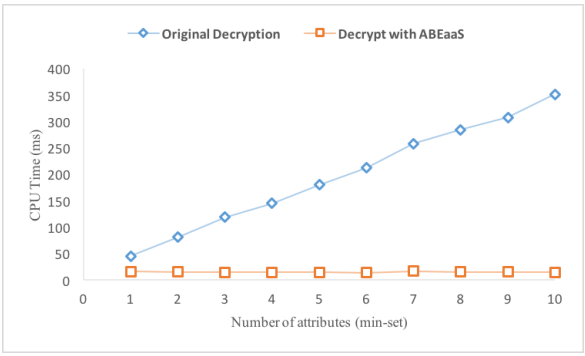
As shown in Figure 5(b), with the growth of the number of attributes, the advantage of using ABEaaS research close to 100

VII. Related Works

Sahai and Waters [1] first proposed the fuzzy Identity-based Encryption, which is the original prototype of Attribute-based Encryption (ABE). Then various works [3]–[5], [14]–[16] is presented to tackle different issues in the ABE. For instance, Ciphertext Policy Attribute based Encryption (CP-ABE), which is proposed by Bethencourt et al. [4], makes ABE scheme adoptable for outsourced data sharing in the cloud environment with features like data confidentiality and access control. Waters [3] also presents another CP-ABE scheme to fix the efficiency and access policy expressiveness issues in previous ABE constructions. The construction from Rouselakis in [5] tackles the issue of pre-defining the attribute



(a) Comparison of users' encryption time



(b) Comparison of users' decryption time

Fig. 4. Comparison of users' operation time (Original ABE scheme v.s. ABEaaS scheme) in General Setting

sets to make the ABE support large universe attribute sets. Even though the efficiency of ABE schemes has been increased, that is not enough. Due to the limitation of computation and battery power of mobile devices, these ABE schemes are still not adaptable in the real mobile cloud scenario.

To tackle the challenge of making ABE available to mobile application, several schemes has been proposed. As the computation delegation [10]–[12] technique has been widely used, computation outsourcing has been introduced into ABE schemes [2], [6]–[9], [13]. Among them, Hohenberger and Waters [2] focus on the "online/offline" encryption to reduce the burden of encryption algorithm in ABE. Green et al. [6] propose the scheme supporting outsourced decryption. To verify the result of outsourced decryption, works in [7] and [8] present different solutions. However, these schemes still stop at the theoretical design phase. Our work will be a beneficial supplement in applying and promoting ABE in the real scenarios.

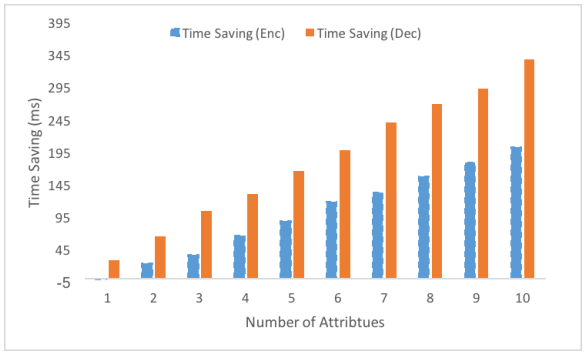
VIII. Conclusion

In this paper, we proposed a novel Attribute Based Encryption as a Service to reduce the computational burden of users' and make the service available over the Internet. We presented the proposed architecture of the ABEaaS. The ABEaaS could be deployed in any cloud environment and provides two setting regarding to different service demands. Furthermore, we also presented the unified algorithms for the ABEaaS instance.

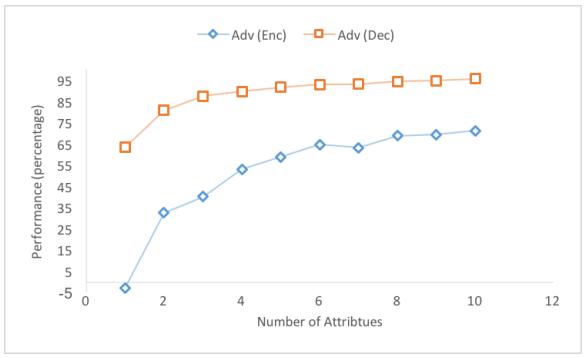
TABLE I
USER'S COMPUTATION ESTIMATES

Schemes	ABE [5]	ABEaaS
Encryption	$\mathbb{B} + (5\|P\| + 2)\mathbb{E} + (2\|P\| + 1)\mathbb{M}$	$\|P\|\mathbb{M}$
Decryption	$(\|P^{'}\| + 2)\mathbb{B} + 2\|P^{'}\|\mathbb{E} + (2\|P^{'}\| + 2)\mathbb{M}$	$\mathbb{M} + \mathbb{E}$

[1] Let $\mathbb{B}$, $\mathbb{E}$ and $\mathbb{M}_p$ be the bilinear map, exponentiation, and multiplication operations, respectively.
[2] Let $|P|$ and $|P^{'}|$ be the complexity of the access policy and the size of the minimal set of attributes, respectively.



(a) Users' time saving using the ABEaaS



(b) P_{Adv} result of the experiment

Fig. 5. Advantage of using the ABEaaS in General Setting

Finally, we presented the security and performance analysis. As future work, we plan to implement and deploy the proposed architecture in the real environment to find the potential system boundary due to number of concurrent users.

ACKNOWLEDGMENT

This work has been supported by 2015 CAE Cybersecurity Research Grant BAA-003-15.

REFERENCES

[1] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.
[2] Susan Hohenberger and Brent Waters. Online/offline attribute-based encryption. In *Public-Key Cryptography–PKC 2014*, pages 293–310. Springer, 2014.
[3] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography–PKC 2011*, pages 53–70. Springer, 2011.
[4] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 321–334. IEEE, 2007.
[5] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 463–474. ACM, 2013.
[6] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security Symposium*, volume 2011, 2011.
[7] Suqing Lin, Rui Zhang, Hui Ma, and Mingsheng Wang. Revisiting attribute-based encryption with verifiable outsourced decryption. *Information Forensics and Security, IEEE Transactions on*, 10(10):2119–2130, 2015.
[8] Baodong Qin, Robert H Deng, Shengli Liu, and Siqi Ma. Attribute-based encryption with efficient verifiable outsourced decryption. *Information Forensics and Security, IEEE Transactions on*, 10(7):1384–1393, 2015.
[9] Sourya Joyee De and Sushmita Ruj. Decentralized access control on data in the cloud with fast encryption and outsourced decryption. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.
[10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology–CRYPTO 2010*, pages 465–482. Springer, 2010.
[11] Kai-Min Chung, Yael Kalai, and Salil Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Advances in Cryptology–CRYPTO 2010*, pages 483–501. Springer, 2010.
[12] Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.
[13] Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *Advances in Cryptology–EUROCRYPT 2012*, pages 318–335. Springer, 2012.
[14] Allison Lewko and Brent Waters. Unbounded hibe and attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 547–567. Springer, 2011.
[15] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *Advances in Cryptology–ASIACRYPT 2012*, pages 349–366. Springer, 2012.
[16] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. Acm, 2006.
[17] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.