A Multi-Cloud based Privacy-Preserving Data Publishing Scheme for the Internet of Things

Lei Yang, Abdulmalik Humayed, and Fengjun Li
The University of Kansas
{lei.yang,ahumayed,fli}@ku.edu

ABSTRACT

With the increased popularity of ubiquitous computing and connectivity, the Internet of Things (IoT) also introduces new vulnerabilities and attack vectors. While secure data collection (i.e. the upward link) has been well studied in the literature, secure data dissemination (i.e. the downward link) remains an open problem. Attribute-based encryption (ABE) and outsourced-ABE has been used for secure message distribution in IoT, however, existing mechanisms suffer from extensive computation and/or privacy issues. In this paper, we explore the problem of privacy-preserving targeted broadcast in IoT. We propose two multi-cloud-based outsourced-ABE schemes, namely the parallel-cloud ABE and the chain-cloud ABE, which enable the receivers to partially outsource the computationally expensive decryption operations to the clouds, while preventing user attributes from being disclosed. In particular, the proposed solution protects three types of privacy (i.e., *data*, *attribute* and *access policy* privacy) by enforcing collaborations among multiple clouds. Our schemes also provide delegation verifiability that allows the receivers to verify whether the clouds have faithfully performed the outsourced operations. We extensively analyze the security guarantees of the proposed mechanisms and demonstrate the effectiveness and efficiency of our schemes with simulated resource-constrained IoT devices, which outsource operations to Amazon EC2 and Microsoft Azure.

1. INTRODUCTION

Today's computing technologies and ubiquitous connectivity have led to a pervasive deployment of intelligence into our daily life in areas as diverse as healthcare (e.g., wearable fitness tracking, remote patient monitoring), home automation (e.g., smart thermostat, security monitoring), smart grid (e.g., load balancing, smart pricing), smart cities (e.g., smart traffic control, distributed pollution monitoring), etc. With more than 20 billions of devices to be connected to the Internet by 2021, as forecast by Cisco and Ericsson [8], the Internet of Things (IoT) materializes a vision of a future Internet that utilizes the sensing and computing capabilities of various devices to facilitate interaction of humans with environment. This has led to the emergence of new categories of applications with impact across industries, businesses, and ultimately, end users.

ACSAC '16, December 05 - 09, 2016, Los Angeles, CA, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4771-6/16/12. . . $15.00
DOI: http://dx.doi.org/10.1145/2991079.2991127

The charm of IoT lies in its capability of interconnecting various sensing devices with varying computation and communication capacity to the Internet. For example, connecting millions of sensors and smart meters to the utilities and power plants makes the aging power grid "smart". The two-way communication capability not only facilitates utilities to collect a variety of real-time, fine-grained data (e.g., power consumption, voltage, phase angle) from smart meters, but also enables them to publish important messages (e.g., control commands, dynamic prices) to smart devices [23]. Similarly, the success of wearable devices and e-health depends not only on the real-time data sensing and reporting, but also on the timely reaction (e.g., firmware upgrade, parameter adjustment) and value-added services (e.g., customized healthy living tips, targeted advertisements) from healthcare service providers and third parties [3]. For example, consider the high risk of heart attacks in extremely hot weather, it is a desirable feature for a wearable device manufacturer to send control messages to devices to increase the measurement frequency of blood pressure and oxygen saturation for elder users with heart diseases and overweight issues.

Nevertheless, with all devices connected and using the Internet infrastructure for data exchange, IoT and its two-way communication expose a new attack surface to adversaries, which makes it susceptible to various security and privacy issues. Consider the tremendous loss due to system failures and attacks, such as power outage and patient data breach, security and privacy has become one of the most important aspects in the design and deployment of IoT applications. Various schemes have been proposed to secure IoT communication and data exchange, which span over a variety of topics including but not limited to privacy-preserving data collection [18, 21], detection and prevention of false data injection [17, 20, 14, 27], patient monitoring [25, 22], etc. However, most of the attention has been concentrated on secure data collection, which considers only one direction (i.e., the *upward* link) of the two-way communication. Little work has been done to secure the other direction (i.e., the *downward* link), where messages are pushed to millions of end devices, neglecting the fact that data along this direction sometimes contains sensitive information, such as system parameters, user-specific prices and healthcare information, and thus needs an enhanced protection.

Conventionally, to securely publish data to a specific target group, the sender (e.g., a control center in smart grid or a value-added service provider in an e-health system) sends data in an encrypted form to each device in a unicast manner. Given the huge number of devices in IoT applications, this will incur a large communication and computation overhead [13] as well as a high complexity for key management. To alleviate the overhead, approaches based on broadcast encryption have been proposed to encrypt the message for an arbitrary set of receivers so that only members of the target group can recover its content. However, solutions based on conventional broadcast encryption [10, 7] require the sender to store key materials for all recipients and incur a storage requirement for the sender. This makes them impractical for IoT applications with a huge number of receiving devices. Besides, in the conventional broadcast encryption, all members in the broadcast set need to be notified of the change [7], which incurs a non-negligible communication overhead to support the incremental growth of the number of users. Therefore, neither option is particularly appealing in the context of IoT.

A more promising solution is to encrypt the message according to common characteristics that specify a set of users as the qualified recipients. For example, recipients can be categorized by their geographical locations such as "*users in San Francisco*". Bethencourt et al. proposed the concept of ciphertext policy (CP-ABE) [4], which is conceptually similar to role-based access control but more fine-grained and flexible. In the CP-ABE model, a user is associated with a set of attributes reflected in her secret key and the access policies, which define the attributes that an authorized user should own, are embedded in the ciphertext. The sender encrypts a message under an access policy so that only authorized users with attributes satisfying the access policy can decrypt the message. The ABE schemes enable a sender to define dynamic access policies without knowing specific receivers in the system beforehand, and thus suit the IoT scenario to enforce fine-grained access control on a large number of receivers. However, a major drawback of ABE is the computational cost for decryption, which increases linearly with the complexity of the access policy (i.e., the number of attributes), due to the expensive paring operations when the receiver matches her attributes to the access policy. As tested in [13], it took about 30 seconds for a smart phone (in particular an iPhone 3G in that experiment) to decrypt a ciphertext containing 100 attributes. For resource-constrained recipients that are typical in an IoT application, the computational cost is too expensive to afford. With the emergence of cloud computing infrastructures such as Amazon EC2 and Microsoft Azure, outsourced-ABE schemes have been proposed [13, 29, 15, 19] to leverage the computational power of the clouds by outsourcing a part of the expensive decryption operations to them.

However, privacy becomes a critical concern when outsourcing decryption operations to a cloud. Most of the existing outsourced-ABE approaches assume that the cloud is fully trusted to host all attributes of a user. In the real world, a cloud server is rarely fully trusted by both the sender and the recipient, especially when user attributes involve multiple sensitive categories. For example, a cloud that is trusted to process attributes such as "*age*" and "*location*" may not be able to access attributes related to users' health status, such as "*having heart disease X*" and "*weight>Y*". Therefore, it is more reasonable to assume a cloud as *honest-but-curious*, that is, it follows the protocol honestly but tries its best to infer users' private information. Specifically, there are three major privacy concerns: (1) *Data privacy.* The sender, e.g., value-added service providers, requires its messages and services to be only accessible to a group of qualified users. Neither the cloud nor other unauthorized users should be able to access the published data. (2) *Attribute privacy.* As explained in the above example, attributes contain sensitive information about the user and thus should be protected as much as possible from being disclosed to the cloud. And (3) *Access policy privacy.* Access policies containing information about data to be published, data sender and data recipient also need to be protected. A stronger privacy requirement is to hide the access policy to avoid privacy inference attacks. We argue that the importance of three types of privacy can be ranked as: data privacy > attribute privacy > access policy privacy.

In general, outsourcing the decryption operations to the cloud is a promising technique to publish information to a group of resource-constrained devices in IoT applications. However, protecting the three types of private information in the targeted broadcasting is still challenging. To address the privacy issues discussed above, we propose two novel CP-ABE schemes that employ multiple clouds to collaboratively complete the outsourced operations, namely *parallel-cloud CP-ABE* and *chain-cloud CP-ABE*. Our schemes delegate user attributes and the decryption operations to multiple clouds, and coordinate them to translate an ABE ciphertext into an ElGamal-type ciphertext, without revealing the original message, the accurate set of user attributes, or the complete access policy to the clouds.

Besides protecting the three types of privacy, our schemes also provide two additional features that are desirable in the IoT data publishing scenario: (i) *Delegation verifiability.* When we outsource the operations for matching attributes to the access policy to the cloud, it is important for the receiver to be able to verify the correctness and completeness of the messages processed by the cloud. For example, a "cheating" cloud server may violate the honest-but-curious assumption and discard messages without performing the matching operation to save its own communication and computation resources. Therefore, the delegation verifiability can be considered as a security enhancement for IoT data publishing applications. And (ii) *Lightweight operations on receivers.* Since the receivers are resource-constrained devices, the operations of decryption and verification at the receiver end are expected to be kept light-weight, which is supported in our solution.

The main contributions of this work are as follows:

- To the best of our knowledge, our work is among the first to protect user attributes against cloud service providers in an outsourced-ABE setting using multiple clouds.
- We propose two multi-cloud-based, outsourced-ABE schemes. The parallel-cloud scheme provides a better performance with strong privacy protection but less system flexibility and less expressiveness for access policies, while the chain-cloud scheme supports flexible customization and expressive access policies at the cost of processing latency.
- We present a lightweight mechanism that allows users

to efficiently verify the correctness and completeness of the partial decryption in clouds.

- Our schemes enable a new application for data publishing in IoT, namely privacy-preserving targeted broadcasting, which is not possible in the past. This new type of data publishing application will benefit not only end users but also information senders such as third-party value-added services providers.
- We thoroughly analyze the security of our proposed schemes and evaluate the performance with experiments using Amazon EC2 and Windows Azure.

2. PRELIMINARIES

2.1 Bilinear Maps

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p, and g be a generator of $\mathbb{G}_0$. e is a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ with the following properties:

1. Bilinearity: $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$.
2. No-degeneracy: $e(g, g) \neq 1$.
3. Computability: for all $u, v \in \mathbb{G}_0$, the bilinear map e is efficiently computable.

2.2 Linear Secret Sharing Scheme (LSSS)

A (k,n)-LSSS shares a secret s over a set of n parties with linear reconstruction property. The secret s is divided to n parties in such a way that any k or more parties can recover the secret and any $k-1$ or less leave the secret completely undermined. Specifically, Shamir's secret-sharing scheme [26] is constructed as:

1. Pick $k-1$ random points to define a polynomial $q(x)$ of degree $k-1$ with $q(0) = s$.
2. Share the secret over n parties by computing $q(i)$ for any $i \in \{1, \cdots, n\}$.
3. Reconstruct the polynomial from any k parties denoted by a set S, using interpolation $q(x) = \sum_{i=1}^{k} q(i)\Delta_{i,S(x)}$. The Lagrange coefficient is $\Delta_{i,S(x)} = \prod_{j\in S, j\neq i} \frac{x-j}{i-j}$, where $S(x)$ denotes an element of S. Finally, recover the secret $s = q(0)$.

2.3 Bloom Filter

Bloom filter (BF) [5] is a space-efficient probabilistic data structure for an approximate representation of a set S, which is typically implemented using a bit-array of w bits with k hash functions. Given an arbitrary element x, a BF supports approximate membership queries "$x \in S$?". It can yield false positive answers but never false negative ones. The probability of false positives can be adjusted by varying w and k, as a tradeoff between space efficiency and the false positive rate.

3. PARALLEL-CLOUD SCHEME

Existing outsourced ABE schemes assume the cloud provider is fully trusted and thus delegate attributes to a single cloud server. However, under the honest-but-curious setting, this assumption is no longer valid. To prevent the cloud server from inferring user privacy from outsourced keys and attributes, we propose a new secure data publishing approach that employs multiple non-colluding cloud servers.

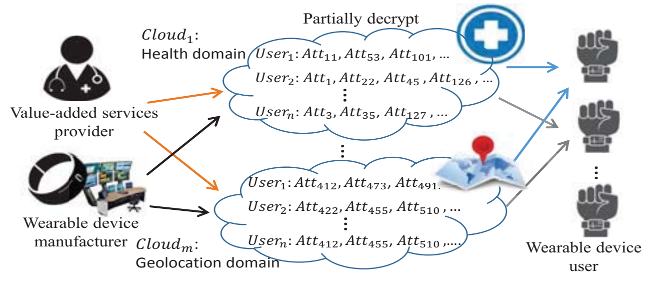

Partially decrypt
$Cloud_1$: Health domain
$User_1$: $Att_{11}, Att_{53}, Att_{101}, \ldots$
$User_2$: $Att_1, Att_{22}, Att_{45}, Att_{126}, \ldots$
$User_n$: $Att_3, Att_{35}, Att_{127}, \ldots$
Value-added services provider
Wearable device manufacturer
$Cloud_m$: Geolocation domain
$User_1$: $Att_{412}, Att_{473}, Att_{491}$
$User_2$: $Att_{422}, Att_{455}, Att_{510}, \ldots$
$User_n$: $Att_{412}, Att_{455}, Att_{510}, \ldots$
Wearable device user

Figure 1: Framework of privacy-preserving targeted broadcast in e-healthcare using parallel-cloud scheme.

We first present a parallel-cloud scheme that divides the attribute set into m parts and outsources each part to one cloud server. The cloud servers operate on the received access structure and ciphertext messages in parallel, and send the intermediate results to the receiver separately. In this process, as long as the cloud servers do not collude with each other, we can protect the complete set of user attributes from any single semi-honest cloud server.

3.1 System Model and Scheme Overview

In this scheme, the system consists of three entities, as shown in Figure 1: the *sender*, such as a value-added service provider or a wearable device manufacturer, publishing a message in the encrypted form; m *cloud servers* partially decrypting the ciphertext; and a large set of targeted device *users* receiving and decrypting the message. In addition, a trusted authority (TA) is implicitly assumed to be in charge of the distribution and management of attributes and private keys to users and cloud servers.

To enforce the fine-grained broadcasting, the sender encrypts a message according to a specific access policy $\mathbb{A}$ in the form of a series of AND gates. For instance, an access structure $\mathbb{A} = DeviceA \wedge HeartDisease \wedge Overweight \wedge CityB$ defines that only the user living in City B who is a customer wearing device A with the heart disease and the overweight issue can decrypt the message. For the simplicity of exposition, we let N denote the universal attribute set of n attributes. The TA divides N into m mutually exclusive subsets as $N = N_1 \cup \cdots \cup N_m$, where $N_i \cap N_j = \emptyset$ for any $i \neq j$, and then outsources each N_i to a cloud server.

For privacy preserving considerations, the attribute splitting and distribution should follow two strategies. First, each subset of attributes is only about one aspect (or domain) of the user. Attributes in any single subset should not provide information for inference attacks. Secondly, the attribute subset is assigned to a cloud considering its service domain and trust level. For example, the cloud server of a private hospital is trusted to host all health-related attributes, while common user attributes such as location can be distributed to a public cloud server.

Correspondingly, the access structure is also divided into m parts such that $\mathbb{A} = \mathbb{A}_1 \cup \cdots \cup \mathbb{A}_m$. When the sender sends the main ciphertext and access structures to cloud servers, each cloud server, on behalf of the receiver, checks if the attributes of the receiver satisfy the access structure. Based on the result of this fine-grained access control, each cloud server decrypts a part of the message and sends it to the user, who will combine the intermediate results received from all the servers to recover the original message.

3.2 Construction

Setup(λ, N, m). The TA calls the Setup algorithm to generate a public key PK and a master key MK. The algorithm takes as input the security parameter λ, a universal attribute set N and the number of cloud servers m. It chooses a bilinear group $\mathbb{G}_0$ of prime order p with a generator g, and the bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. Next, the setup algorithm chooses two randoms $\alpha, \beta \in \mathbb{Z}_p$ and generates PK and MK:

$$PK = (\mathbb{G}_0, g, h = g^{\beta}, e(g,g)^{\alpha}),$$
$$MK = (\beta, g^{\alpha}).$$

KeyGen(MK, S, t). Assume each user has a set of attributes $S = S_1 \cup \cdots \cup S_m$, where $S_1 \subseteq N_1, \cdots, S_m \subseteq N_m$ and $\forall_{i \neq j} S_i \cap S_j = \emptyset$. When a new user (e.g., a wearable device) joins the system, it registers to the TA with S and a random $t \in \mathbb{Z}_p$ chosen by itself. Then, the TA calls the key generation algorithm to prepare a transformation key TK for the clouds to perform partial decryption, from which the user can recover the final message with his private key $SK = t$. In particular, KeyGen selects a random $r \in \mathbb{Z}_p$ and a random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$, and takes S, t, and the master key MK as input to generate the transformation key. TK is set as $TK = (D = g^{t(\alpha+r)/\beta}, D_j = g^r H(j)^{r_j}, D_j{}' = g^{r_j}, \forall j \in S)$. H and H_1 denote collision-free hash functions, where $H : \{0,1\}^* \to \mathbb{G}_0$. Finally, the TA distributes D, D_j, and D'_j to $Cloud_j$.

Encrypt(PK, M, $\mathbb{A}$). To broadcast a message M under the access structure $\mathbb{A}$, the sender first chooses m random numbers $s_1, \cdots, s_m \in \mathbb{Z}_p$, where the secret s_i is shared by all attributes in $\mathbb{A}_i$. Let $k_i = |\mathbb{A}_i|$ be the number of elements in $\mathbb{A}_i$ and $index(y)$ be the index of the attribute y in $\mathbb{A}_i$. To share the secret, a polynomial $q_i(x)$ of degree $k_i - 1$ is constructed for $\mathbb{A}_i$, where $q_i(0) = s_i$ and the other $k_i - 1$ values are randomly set to complete the construction. Given $\mathbb{A}_i$, the ciphertext CT is then constructed as:

$$\widetilde{C} = Me(g,g)^{\alpha s}, C = h^s, CT_i$$

where $s = \sum s_i$, and $CT_i =$

$$\mathbb{A}_i, C_y = g^{q_i(index(y))}, C'_y = H(y)^{q_i(index(y))}, \forall y \in \mathbb{A}_i$$

To allow the receiver to verify the correctness and completeness of the transformation performed by the clouds, the sender also generates a verification value $V = senderID| H_1(\mathbb{A})|seq(H_1(\mathbb{A}))|T$, where "|" denotes concatenation. $H_1(\mathbb{A})$ is the digest of the access policy $\mathbb{A}$, $seq(H_1(\mathbb{A}))$ is the sequence number of the message regarding the access policy $\mathbb{A}$, and T is the timestamp of the message. The sender maintains the sequence number for each access policy, and increases its value by 1 when sending a new message under that access policy. Then, the sender generates the signature of message M and verification value V as $\sigma = sign(H_1(M|V))$. Finally, the sender randomly chooses a cloud, e.g., $Cloud_1$, to hold $(\widetilde{C}, C, \sigma)$, and sends CT_i and V to $Cloud_i$.

Transform(CT_i, TK_i). When a cloud, e.g., $Cloud_i$, receives CT_i, it uses the transformation key TK_i to partially decrypt the ciphertext and transforms it into a form whose decryption is less computationally costly. In particular, the cloud first checks if $\mathbb{A}_i \subseteq S_i$. If not, it returns an error symbol $\perp$, indicating that the user does not satisfy the access structure. If $\mathbb{A}_i \subseteq S_i$, for each attribute $j \in \mathbb{A}_i$, it computes:

$$\begin{aligned} DecryptNode(CT_i, TK_i, j) &= \frac{e(D_j, C_j)}{e(D'_j, C'_j)} \\ &= \frac{e(g^r H(j)^{r_j}, g^{q_i(index(j))})}{e(g^{r_j}, H(j)^{q_i(index(j))})} \\ &= e(g,g)^{rq_i(index(j))} \end{aligned}$$

After the cloud computes the values for all attributes in $\mathbb{A}_i$, it combines them to partially recover the secret s_i that is shared in $\mathbb{A}_i$. In particular, it computes $F(\mathbb{A}_i)$ as below and sends the result together with V to the receiver:

$$\begin{aligned} F(\mathbb{A}_i) &= \prod_{j \in \mathbb{A}_i} (e(g,g)^{rq_i(index(j))})^{\Delta_{j,\mathbb{A}_i}(0)} \\ &= e(g,g)^{r \sum_{j \in \mathbb{A}_i} q_i(index(j)) \cdot \Delta_{j,\mathbb{A}_i}(0)} \\ &= e(g,g)^{rq_i(0)} \\ &= e(g,g)^{rs_i} \end{aligned}$$

With C and D, $Cloud_1$ computes $\widetilde{D}$ as:

$$\begin{aligned} \widetilde{D} &= e(C, D) \\ &= e(g^{\beta s}, g^{t(\alpha+r)/\beta}) \\ &= e(g,g)^{st(\alpha+r)} \end{aligned}$$

Finally, it sends the partially decrypted ciphertext $\widetilde{C}$, $\widetilde{D}$ and σ to the receiver.

Decrypt($\widetilde{C}, \widetilde{D}, F(\mathbb{A}_i), SK$). If the receiver receives m parts of partial ciphertexts, she knows that her attributes satisfy the access policy. Otherwise, she discards the partial ciphertext without decryption.

With its private key SK and the ciphertext transformed by the clouds (i.e., $\widetilde{D}$ and $F(\mathbb{A}_i)$s), the receiver recovers the original message as:

$$\begin{aligned} \frac{\widetilde{C}}{\frac{(\widetilde{D})^{1/t}}{\prod_{i=1}^{m} F(\mathbb{A}_i)}} &= \frac{\widetilde{C}}{\frac{(e(g,g)^{st(\alpha+r)})^{1/t}}{\prod_{i=1}^{m} e(g,g)^{rs_i}}} &= \frac{\widetilde{C}}{\frac{e(g,g)^{s(\alpha+r)}}{e(g,g)^{rs}}} \\ &= \frac{Me(g,g)^{\alpha s}}{e(g,g)^{\alpha s}} &= M. \end{aligned}$$

In the above computation, it is obvious that the receiver does not need any paring operation to recover the message. Instead, it only takes one exponentiation and m multiplication operations regardless of the complexity of the access structure. Compared to the original CP-ABE, which requires $2 \sum |\mathbb{A}_i|$ pairings, our scheme greatly reduces the computational overhead at the receiver.

Verify(M, σ, V). To verify the completeness of the operations done by the clouds, the receiver needs to check if all the verification values are consistent. As long as at least one cloud is honest, the received verification data is authentic. Then, the receiver randomly chooses a V from m partial ciphertexts, and uses $senderID$ and $H_1(\mathbb{A})$ to look up the sequence number $seq_{prv}(H_1(\mathbb{A}))$ of the previous round for $H_1(\mathbb{A})$ from $senderID$. The receiver verifies the completeness, that is, she has received all messages intended to her, by checking if $seq(H_1(\mathbb{A})) = seq_{prv}(H_1(\mathbb{A})) + 1$. If not, this indicates some messages are discarded by dishonest cloud(s). After the completeness verification, the receiver checks $M|V$ against the signature σ to verify the correctness of the recovered message, and updates $seq(H_1(\mathbb{A}))$.

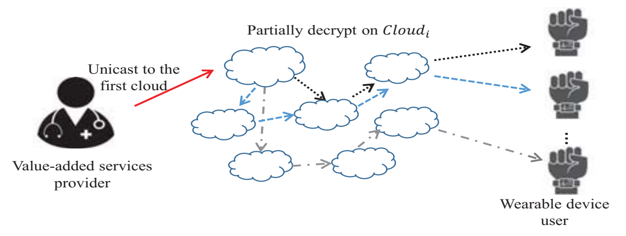

Partially decrypt on $Cloud_i$
Unicast to the first cloud
Value-added services provider
Wearable device user

Figure 2: Framework of privacy-preserving targeted broadcast in e-healthcare using chain-cloud scheme.

Discussions. The parallel-cloud scheme hides the complete set of user attributes from a single cloud server, and significantly reduces the computational overhead at the receiver. However, it yields three drawbacks. First, the scheme only supports the AND gate in the access structure. Thus, it is less expressive comparing to the monotone structure proposed in the original CP-ABE design [4], which supports k-threshold gates. Secondly, the scheme uses the number of clouds, m, as a system-wide parameter, which makes the system structure very rigid. It requires all the receiver to use the same value for m, and imposes an additional burden to the sender, who is required to split the access structure into m pieces and distribute to m cloud servers. As individual users may have different needs regarding the protection of the attribute privacy, schemes that provide a flexible privacy setting are more desirable. Finally, the parallel-scheme incurs unnecessary communication between a cloud and the receiver. For example, when $Cloud_i$ successfully matches $\mathbb{A}_i$ to S_i but $Cloud_j$ fails to match $\mathbb{A}_j$, $Cloud_i$ has to partially decrypt the message and forward the intermediate result to the receiver, regardless of the fact that the message cannot be decrypted by the user eventually. Frequent partial matchings incur an increasing computation and communication overhead to the clouds. To overcome these drawbacks, we propose an enhanced scheme using a chain-cloud structure and present it in the next section.

4. CHAIN-CLOUD SCHEME

4.1 Overview and System Model

Unlike the parallel-cloud scheme, which requires the senders to connect to a same m cloud servers simultaneously, the chain-cloud scheme allows each receiver to specify how many clouds to use and how attributes are delegated to each cloud. As shown in Figure 2, three receivers choose three sets of clouds to form three different paths to three devices. Moreover, the chain-cloud scheme allows the sender to encrypt a message under an expressive access policy $\mathbb{A}$ with k-threshold gates. That is, an access policy can be satisfied with any k or more attributes.

To support this, we employ a Bloom filter [5] in each cloud to hold the part of attributes delegated to it. In particular, when a cloud receives an encrypted message, it first checks if the attributes that are delegated to it satisfy the access structure: if so, the cloud server partially decrypts the message and sends the result to the receiver; otherwise, it further looks up all the attributes in the access policy against the Bloom filter to check if the attributes have a chance to satisfy the access structure. If the Bloom filter look-up returns a positive result, it indicates that it is possible the receiver has

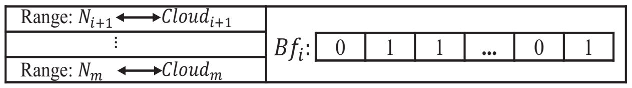


Figure 3: The Bloom filter sent to each cloud.

the required attribute(s) to satisfy the access structure. So, the cloud should decrypt the attributes in the access structure as many as possible and forward the partially decrypted message to a next cloud. If the Bloom filter returns a negative result, which means the user does not have the required attribute(s) for sure, the cloud should stop propagating the message. Starting from the first cloud, this process repeats until either a cloud in the path finds that the access structure cannot be satisfied, or the partially decrypted message is successfully forwarded to the receiver.

4.2 Construction

Setup(λ, N). The setup algorithm is similar to the one in the parallel-cloud scheme, except that it does not require a specific value for m, i.e., the number of clouds to be used.

BloomFilterGen$(S, [Cloud_i])$. In the parallel-cloud scheme, the user (i.e., receiver) decides the clouds to be used and calls the BloomFilterGen algorithm, which takes the user's attribute set S and the number of clouds m as input to generate m Bloom filters. Therefore, the user has a full control in deciding how to delegate her attributes to multiple clouds and in what order, by taking the sensitiveness of her attributes and her trust to cloud service providers into account. This not only provides a flexible mechanism for the user to determine her own multi-cloud platform setting, but also fits the real-world cloud usage scenario. That is, health-related attributes are likely to be stored in a private hospital cloud (and thus more trusted), while profile attributes can be delegated to public clouds.

The selected m clouds are organized as a chain. Each cloud delegates a subset of attributes (S_i), from which the Bloom filter is generated. Here, we use an example to explain the generation process. Suppose the size of the universal attribute set N is 200 and the user selects three clouds, where $Cloud_1$ is responsible for the attributes in $S_1 \subseteq [1, 100]$, $Cloud_2$ for $S_2 \subseteq [101, 160]$ and $Cloud_3$ for $S_3 \subseteq [161, 200]$. To build the Bloom filter BF_1 for $Cloud_1$, the attributes $j \in S_2$ and $j' \in S_3$ are inserted into the BF_1. We can adjust the false positive rate by changing the size of the Bloom filter and the number of hash functions. To further increase the probability of false positives, we randomly set some bits in the filter to 1 to introduce noise. As shown in Figure 3, our Bloom filer consists of two parts. The first part is a lookup table used by a cloud to locate the next cloud, which is in charge of a specific range of attributes, and the second part is the noisy Bloom filter, which contains attributes in $Cloud_{i+1}, \cdots, Cloud_m$, and the noise bits.

In the chain-cloud scheme, each user has her own cloud usage specification (CUS) for attribute delegation and can change it anytime without informing the senders. This makes it more flexible than the parallel-cloud scheme. Then, the user sends the cloud usage specification to the TA to correctly distribute the corresponding transformation keys to the clouds in use.

KeyGen(MK, S, t, CUS). The KeyGen algorithm takes as input the master key, a user's attribute set S and private key t, and her cloud usage specification CUS. It generates the private key SK and the transformation key TK in the same way as described in Section 3.2. Then, the TA sends the corresponding transformation key TK to each cloud according to the CUS.

Encrypt(PK, M, $\mathbb{A}$). The chain-cloud scheme can support expressive access policies such as access tree defined in the original CP-ABE [4]. Here, we take access tree as an example to briefly explain the encryption algorithm.

Let T be the access tree representing the access structure $\mathbb{A}$. Each non-leaf node x of the tree is a threshold gate associated with a threshold value k_x, where $0 < k_x \leq num_x$ for a node with num_x children. A leaf node is associated with an attribute and a threshold $k = 1$. We index the children of each node from 1 to num_x, and use $index(x)$ to return the index value of x among its sibling nodes.

Upon receiving an encrypted message, the cloud checks if the access tree is satisfied. Let T_x denote a subtree rooted at node x. If a set of attributes γ satisfies the access tree T_x, we denote it as $T_x(\gamma) = 1$. $T_x(\gamma)$ can be computed recursively: for a leaf node x, $T_x(\gamma)$ returns 1 if the attribute of x is in γ; for a non-leaf node y, the cloud evaluates $T_{y'}(\gamma)$ for all the children $y\prime$ of node y, and sets $T_y(\gamma)$ to 1 if at least k_y children return 1.

To encrypt a message, the Encrypt() algorithm first chooses a polynomial q_x in a top-down manner, starting from the root R, for each node x in the tree. For each node x in the tree, the degree d_x of the polynomial q_x is set to $k_x - 1$, where k_x is the threshold. For the root R, the algorithm chooses a random secret $s \in \mathbb{Z}_p$ and sets $q_R(0) = s$. Then, it chooses d_R additional random values to construct q_R completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$, where $parent(x)$ denote the parent node of x, and chooses d_x additional randoms to define q_x.

Let Y be the set of leaf nodes in $\mathbb{A}$. Once the access tree is defined, the algorithm encrypts the message as below and sends the the ciphertext to the first cloud server in the chain.

$$\begin{aligned} CT = (&\mathbb{A}, \widetilde{C} = Me(g,g)^{\alpha s}, C = h^s, \\ &\forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(y)^{q_y(0)}). \end{aligned}$$

Transform(CT, TK_i). When $Cloud_i$ receives the partially decrypted ciphertext from the previous cloud in the chain, it calls Transform() for further decryption.

For a leaf node x with attribute $j \in N_i$ and $j \in S_i$, $Cloud_i$ calls $DecryptNode()$, which is described in Section 3.2, and computes $\frac{e(D_j, C_j)}{e(D'_j, C'_j)} = e(g,g)^{rq_j(0)}$. The node is then marked as $satisfied$ and the node value is set to $e(g,g)^{rq_j(0)}$.

If $j \in N_i$ but $j \notin S_i$, which means attribute j is in charge of $Cloud_j$ but does not belong to the attribute set S_j of the user, we should set the node value to an error symbol $\perp$, indicating *unsatisfied*.

If $j \notin S_i$ but $j \notin N_i$, the attribute j (i.e., a leaf node) may be outsourced to another cloud. So, $Cloud_i$ looks it up in the Bloom filter and sets the node to $\perp$ if the result is negative. Note that, a positive result cannot guarantee the ownership of the attribute due to the false positive introduced by the Bloom filter. Also, not all the leaf nodes are associated with an attribute. Some are the internal nodes before the transformation done by the previous cloud.

We then compute the value for the non-leaf nodes in a bottom-up manner. For a non-leaf node x, we choose an arbitrary k_x-sized set E_x of its child nodes z to test if it satisfies node x. If such set exists, we compute the value of x as V_x:

$$\begin{aligned} V_x &= \prod_{z \in E_x} V_z^{\Delta_{i,E'_x}(0)}, \textit{where } i{=}index(z) \textit{ and } E'_x{=}\{index(z){:}z{\in}E_x\} \\ &= \prod_{z \in E_x} (e(g,g)^{rq_z(0)})^{\Delta_{i,E'_x}(0)} \\ &= \prod_{z \in E_x} (e(g,g)^{r \cdot q_{parent(z)}(index(z))})^{\Delta_{i,E'_x}(0)} \\ &= \prod_{z \in E_x} e(g,g)^{r \cdot q_x(i) \cdot \Delta_{i,E'_x}(0)} \\ &= e(g,g)^{r \cdot q_x(0)}. \end{aligned}$$

Otherwise, the value of x is set to $\perp$. Repeatedly, we compute the value of the root node R as $V_R = e(g,g,)^{rq_R(0)} = e(g,g)^{rs}$.

Next, $Cloud_i$ computes $\widetilde{D} = e(g,g,)^{st(\alpha+r)}$ as described in Section 3.2, and forwards $\widetilde{C}, \widetilde{D}, V_R$ to the receiver. If the computation stops before arriving at root R but the Bloom filter indicates that an attribute satisfying the access structure may exist in subsequent cloud servers, $Cloud_i$ looks up the table in Figure 3 and forwards the intermediate result to the next cloud that probably hold the attributes to satisfy the entire tree. If no such cloud exists, it stops propagating and thus reduces the computation and communication overhead in the subsequent clouds. In this way, the noisy Bloom filter designed for the chain-cloud scheme increases the transformation efficiency. Only when an access policy can probably be satisfied by certain subsequent clouds, the current cloud forwards the message.

As shown in the example in Figure 4, when $Cloud_1$ finds that the access tree is satisfied by itself and $Cloud_3$, it skips $Cloud_2$ and directly forwards the intermediate result to $Cloud_3$.

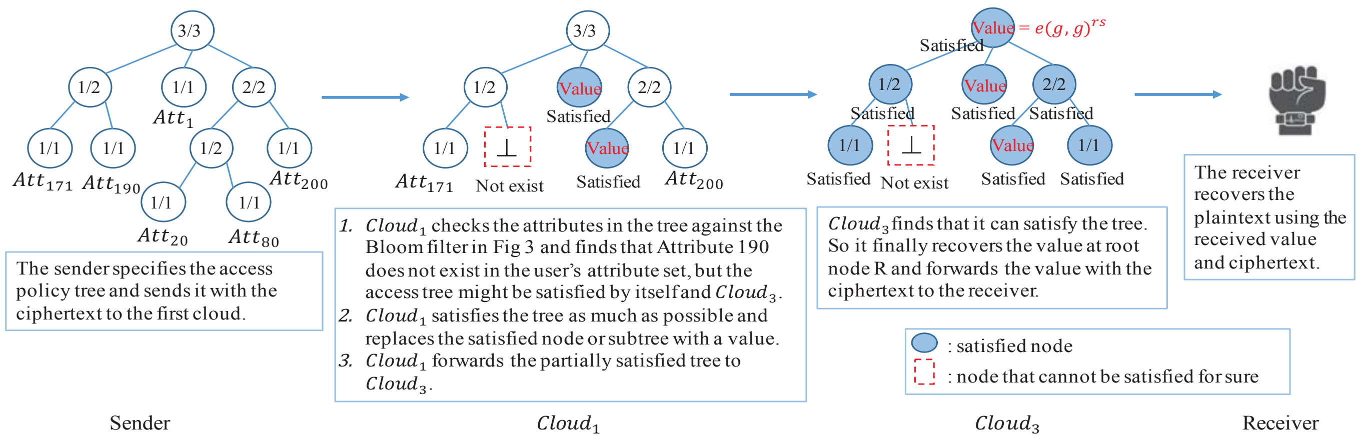
Decrypt($\widetilde{C}, \widetilde{D}, V_R, SK$). Once receiving the partially decrypted message from a cloud server, the receiver performs the final decryption operation as below.

$$\begin{aligned} \frac{\widetilde{C}}{\frac{(\widetilde{D})^{1/t}}{V_R}} &= \frac{\widetilde{C}}{\frac{(e(g,g)^{st(\alpha+r)})^{1/t}}{e(g,g)^{rs}}} &&= \frac{\widetilde{C}}{\frac{e(g,g)^{s(\alpha+r)}}{e(g,g)^{rs}}} \\ &= \frac{Me(g,g)^{\alpha s}}{e(g,g)^{\alpha s}} &&= M. \end{aligned}$$

Obviously, the decryption overhead on resource-constrained devices is reduced to one exponentiation operation in the chain-cloud scheme.

Verify(M, σ, V). The verification algorithm of the chain-cloud scheme is similar to the one in the parallel-cloud scheme discussed in Section 3.2. The sender counts the message encrypted under the access policy $\mathbb{A}$, generates the verification data $V = senderID|H_1(\mathbb{A})|seq(H_1(\mathbb{A}))|T$, and signs the message and verification data as $\sigma = sign(H_1(M|V))$.

Once the receiver recovers the plaintext M and the verification value V, she first checks them against the signature σ to verify the correctness of the transformation, and then verifies the completeness by comparing if the received sequence number is greater than the stored sequence number by 1, for the access policy $\mathbb{A}$.



The sender specifies the access policy tree and sends it with the ciphertext to the first cloud.

1. $Cloud_1$ checks the attributes in the tree against the Bloom filter in Fig 3 and finds that Attribute 190 does not exist in the user's attribute set, but the access tree might be satisfied by itself and $Cloud_3$.
2. $Cloud_1$ satisfies the tree as much as possible and replaces the satisfied node or subtree with a value.
3. $Cloud_1$ forwards the partially satisfied tree to $Cloud_3$.

$Cloud_3$ finds that it can satisfy the tree. So it finally recovers the value at root node R and forwards the value with the ciphertext to the receiver.

The receiver recovers the plaintext using the received value and ciphertext.

Figure 4: A transformation example: attributes set[1,100] is outsourced to $Cloud_1$, [101, 160] to $Cloud_2$, and [162,200] to $Cloud_3$.

5. SECURITY ANALYSIS AND PERFORMANCE EVALUATION

In this section, we first analyze the security features of the proposed multi-cloud ABE schemes, and then explain our implementation of the two schemes on Amazon EC2 and Microsoft Azure.

5.1 Security Analysis

Data Privacy. The sender does not want the clouds or other unauthorized parties to access the message it sends to a target group of users. From the decryption algorithms in Section 3.2 and Section 4.2, we see that an adversary needs to be able to cancel out $e(g,g)^{\alpha s}$ from the ciphertext $\widetilde{C}$ to recover the plaintext message. In doing so, he needs to be able to compute the pairing value over C from the ciphertext and D from the transformation key, respectively, to cancel out the secret $e(g,g)^{rs}$. The security of the pairing operation ensures that an unauthorized party without knowing the correct r and s cannot recover this secret. Meanwhile, since the paring value is blinded by the private key t in our scheme, it is impossible for a cloud to recover the plaintext, even though it has access to $e(g,g)^{rs}$.

Attribute Privacy. To reduce the computational cost at the devices, the operation of matching the attributes to the access structure is outsourced to the cloud. As the attributes of the user are disclosed to the cloud, it introduces a serious privacy concern, especially when the attributes contain sensitive information about the user. One may argue that in real-world, attributes delegated to the clouds are represented in the form of a hash value, instead of the meaningful raw text, and thus incurs less privacy risk. Actually, a malicious cloud can still launch the dictionary attack to check every possible word against the hash value. Therefore, the attribute privacy is considered unprotected in all the existing outsourced ABE schemes using a single cloud.

In our multi-cloud schemes, each cloud server is only in charge of a part of attributes, so that no single cloud can learn the complete set of attributes of a user. This significantly reduces the privacy leakage caused by the attribute-based inference attacks due to outsourced decryption. To formally assess the improved protection to attribute privacy in our multi-cloud schemes, we define *accuracy* as a measure of the degree of knowledge about a user, and compare with the outsourced ABE schemes using a single cloud. The higher the accuracy, the more the cloud knows about a user.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

where TP, FP, TN, FN represent true positive, false positive, true negative, and false negative, respectively. Specifically, TP denotes the number of attributes that the cloud is certain that a user has, and TN denotes the number of attributes that the cloud is certain that a user does not have. From the definition, we see that the lower bound of accuracy is 0.5, when the cloud has no knowledge about a user and thus can only guess with a probability of 0.5.

Assume the universal attribute set is N, a user's attribute set is S, and the number of clouds is m. In the single-cloud outsourced ABE, the accuracy achieves its upper bound, $(|S| + |N - S|)/|N| = 1$, since the cloud clearly knows that the user owns $|S|$ attributes and does not own the remaining $|N - S|$ attributes.

In the parallel-cloud scheme, each cloud server works independently from other clouds. It knows only the attributes outsourced to itself, i.e., $|S|/m$ attributes. For the remaining attributes, it can only guess with a probability of 0.5. So, the accuracy is at most $(|S|/m + 0.5(|N| - |S|/m))/|N|$. In a setting that $|N| = 200, |S| = 100, m = 5$, the accuracy is 0.55. Obviously, the parallel-cloud scheme improves attribute privacy significantly compared to the single-cloud outsourced ABE.

In the chain-cloud scheme, for the sake of efficiency, each cloud stores a Bloom filter to check if the access policy has a chance of being satisfied by the subsequent clouds. This incurs attribute privacy leakage. However, the degree of leakage is controllable by the user by carefully selecting the cloud and determining its position in the cloud chain. This is because the cloud which is closer to the tail of the chain cannot test the membership of any attribute in the clouds that are closer to the head of the chain. Moreover, the user can further adjust the false positives caused by the Bloom filter by adding noise, changing its size and the number of hash functions. In an extreme case, to prevent a cloud from inferring the attributes in other clouds from the Bloom filter, all positions in a Bloom filter need to be set to 1. With these noise bits, the cloud forwards all the messages that are not satisfied so far, which is actually equivalent to disabling the Bloom filter. As a result, each cloud knows only the attributes it holds.

To measure the accuracy, we need to set p_{FP}, which is the probability of false positive of the Bloom filter, and compute:

$$\frac{\frac{|S|}{m} + |N-S| * (1-p_{FP}) + 0.5(|N| - \frac{|S|}{m} - |N-S| * (1-p_{FP}))}{|N|}$$

where the fist part of the numerator is the true positives of attributes that are outsourced to the cloud, the second part is the true negatives that the cloud gets from the Bloom filter, and the third part is the correct guess of the remaining attributes with the probability of 0.5.

Let us set p_{FP} to 0.6, for example, the accuracy on the fist cloud of the chain, who knows the most information about a user, is 0.65. We see that the cloud gets more accurate information about a user from the chain-cloud scheme than the parallel-cloud scheme. When the probability of false positive is set to 1, it becomes equal to that in the parallel-cloud scheme.

Access Policy Privacy. In the single-cloud outsourced ABE, a cloud server can see the complete access policy and further infer the underlying message. In our parallel-cloud scheme, since an access policy is divided into multiple pieces, each cloud knows only a part of the policy. In the chain-cloud scheme, to support the flexible system structure and the complete expressiveness of the access policy, the access structure is distributed over the chain. Any cloud in the chain has no knowledge about the attributes in previous clouds, since the attributes that are satisfied in previous clouds have been replaced with node values, which looks like a random value. Therefore, a cloud that is closer to the tail sees less about access structure. Besides, the chain is organized in a way that more trusted clouds are placed at the beginning positions, thus it is reasonable to assume that allowing them to see a more access structure will not cause severe privacy leakage.

Collusion Resistance. A major challenge to construct a secure ABE scheme is to prevent colluding users so that they cannot combine their attributes to satisfy an access policy, which they cannot decrypt individually. Our proposed schemes are resistant to the colluding attacks. Similar as the original CP-ABE design, we select a random r for each user in the key generation algorithm, which results in distinct values for different users when recovered by the secret sharing scheme.

Verifiability. Some messages from the sender may include critical content, such as control commands, and thus the correctness verification of the transformation is very important. We use the public-key signature scheme in both schemes to enable end-to-end verification. Since the private key for singing the message is only known to the sender, no cloud nor adversary can forge a valid signature.

Another challenge is to verify the completeness of the transformation. A cloud may accidentally fail in matching an access policy to the attributes, due to system errors or intentional misbehavior. To verify the completeness, the sender and the receiver need to share a common knowledge about how many messages are transmitted. We adopt a stateful verification scheme for completeness, which maintains a continuously increasing counter for the messages that are encrypted under a specific access policy $\mathbb{A}$ as the shared knowledge. Since the counter is also signed by the sender, the receiver can trust it to verify if any message is accidentally or maliciously discarded. However, this completeness verification still has limitations. If a malicious cloud never forwards a message under a specific access structure, which should have been satisfied by the receiver, the receiver cannot build the shared knowledge and know the existence of a message without interacting with the sender. A simple yet effective countermeasure for the parallel-cloud scheme is that the sender sends the complete access policy $\mathbb{A}$ instead of a part $\mathbb{A}_i$ to each cloud and the cloud forwards it to the receiver. The parallel-cloud scheme introduces redundancy for completeness verification. As long as at least one cloud is honest, the receiver can verify if she matches the access policy, and determines if a malicious cloud exists. The receiver can afford such lightweight matching operation as it only needs to compare if two attributes are identical. However, the drawback of this countermeasure is that the cloud knows the complete access policy, which may cause privacy leakage. We argue that this is a reasonable price to pay, considering the criticalness of the completeness. Moreover, completeness verification remains a challenging task for the chain-cloud scheme. Since only the last cloud successfully satisfying the access policy will forward the message to the receiver, it does not provide redundancy as the parallel-cloud scheme does. In fact, the completeness verification is still a challenging task even for the general outsourcing applications such as searchable encryption, which involves only one cloud. To the best of our knowledge, only accumulator [24] can provide the completeness verification at a very high cost, and there is no known solution for outsourced ABE scheme. We consider completeness verification for the chain-cloud outsourced ABE scheme an open problem for our future work.

Scheme	Complexity of decryption
CP-ABE	(2n+1)P + 2$\mathbf{M}_2$
Parallel-cloud CP-ABE	$\mathbf{E}_2$ + 2$\mathbf{M}_2$
Chain-cloud CP-ABE	$\mathbf{E}_2$ + (m+2)$\mathbf{M}_2$

Table 1: Comparison of asymptotic complexity of decryption operation of different scheme.

5.2 Performance Evaluation

We implement the proposed parallel-cloud and chain-cloud outsourced ABE schemes in real-world clouds, i.e., Amazon EC2 and Microsoft Azure, and compare the performance of our schemes with the one of the original CP-ABE [4] in terms of asymptotic complexity and the experimental performance.

Since we do not make changes to the encryption algorithm, and the partial decryption is delegated to the cloud which is assumed to have unlimited computation capability, we focus on the comparison of the overhead at the recipient devices. Note that in the implementation, a message itself is actually encrypted using AES keys, which has fixed computational overhead. So, we only evaluate the overhead introduced by the ABE operations.

Table 1 compares the asymptotic complexity of the three schemes, where $\mathbf{P}$ denotes the paring operation, $\mathbf{E_2}$ denotes the group exponentiation, $\mathbf{M_2}$ denotes the group multiplication in $\mathbb{G}_2$, n denotes the number of attributes in the ac-

	CP-ABE	OP-CP-ABE	OC-CP-ABE
2 clouds	1444	830	1620
5 clouds	1444	732	1931

Table 2: Comparison of the delay between the sending time and the receiving time in the 2-cloud and 5-cloud settings with 60 attributes in the access policy.

cess policy, and m denotes the number of clouds used in the parallel-cloud outsourced ABE. Obviously, the computational complexity of our schemes is independent from the complexity of access policy, which only needs a constant time to recover the plaintext.

Next, we implement the original CP-ABE and our schemes using the Java Pairing-Based Cryptography Library [1]. We use Type A elliptic curve of 160-bit group order, which provides 1024-bit discrete log security equivalently. The experiments are conducted on the Raspberry Pi 2 [2], with 700MHz ARM A6 microprocessor and 512 MB RAM, to simulate the resource-constrained IoT devices such as smart phones acting as the gateway for wearable devices and smart meters. We launch multiple Amazon EC2 and Windows Azure instances to simulate the cloud service providers in our schemes. The partial decryption in the clouds in the parallel-cloud scheme is obviously efficient since all the cloud servers decrypt the corresponding pieces simultaneously. However, for the chain-cloud scheme, since the partial decryption is conducted in a sequential manner, we are interested in evaluating the delay introduced by the chain structure.

Figure 5 compares the decryption time of the original CP-ABE, our proposed parallel-cloud CP-ABE and chain-cloud CP-ABE on the Raspberry Pi 2, using a multi-cloud platform of 5 clouds. Our asymptotic complexity analysis is confirmed by the real implementation, that is, the decryption time of the original CP-ABE is proportional to the complexity of the access structure, while the overhead in our schemes is significantly reduced to a constant value, regardless of the access structure. This shows the benefit of lightweight decryption introduced by the outsourcing.

Finally, we evaluate the delay introduced by the multi-cloud structure. To evaluate the chain-cloud scheme, we connect the instances of EC2 and Azure alternately to measure the delay, since the communication time between two servers from the same cloud service provider (e.g., two Amazon instances) is negligible. The round trip time (RTT) of two EC2 virtual machines (VMs) is less than 2 milliseconds, while the RTT between an EC2 VM and an Azure VM is around 60 milliseconds. The sender and the recipient are synchronized through a Socket communication.

Table 2 compares the delay between the time of sending out the ciphertext and the time of receiving all the partially decrypted ciphertexts from the cloud servers. In this experiment, the message is encrypted under an access policy with 60 attributes. We use the single-cloud outsourced ABE as the base line, and compares the delay in the 2-cloud and 5-cloud settings. We see that the parallel-cloud scheme achieves the best performance in terms of delay, since each cloud partially decrypts a small part of the ciphertext and transmits it in a parallel way. Compared to the base line, the chain-scheme has a larger delay, because all the clouds need to sequentially decrypt the ciphertext, which introduces the transmission delay and the delay caused by the serialization and un-serialization of data for network transmission.

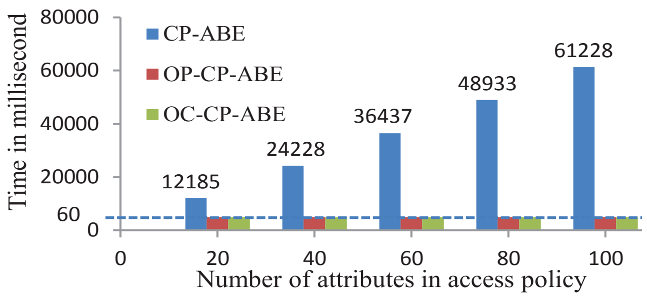


Figure 5: Comparison of the decryption time between the original CP-ABE, the parallel-cloud outsourced CP-ABE and the chain-cloud outsourced CP-ABE with varying number of attributes in the access structure.

6. RELATED WORK

Attribute-based messaging. Extensive studies have been done on secure data publishing. Bobba et al. developed an attribute-based messaging system where senders can dynamically create a list of recipients based on their attributes [6]. However, this scheme incurs high computational overhead and thus only suits for traditional PC-based applications. Fadlullah et al. proposed a secure targeted broadcast scheme for smart grid where the utility encrypts a message using key policy attribute-based encryption (KP-ABE) and broadcasts the ciphertext to a specific group of users [9]. KP-ABE is the first ABE scheme introduced by Sahai and Waters [12]. In KP-ABE, each encrypted message is labeled with a set of attributes and each user is assigned a private key associated with an access structure. A user decrypts an encrypted message only when the attributes associated with the ciphertext satisfy her access policy. Compared to CP-ABE, KP-ABE is less expressive in specifying who has access to the encrypted message. Fadlullah's ABE-based broadcast exploits the original KP-ABE construction without outsourcing the decryption operation. Although it avoids the problem of attribute privacy leakage, it imposes a very large computational overhead on receivers, which is prohibitively high for resource-constrained devices. In [15], the authors proposed a practical ABE-based data sharing scheme using CP-ABE and the outsourcing technique to reduce the overhead on smart meters. However, this scheme is still susceptible to the privacy leakage risk, that is, when a user delegates all the attributes to a single cloud, the cloud is able to infer her sensitive information from the attributes.

Verifiable outsourced attribute-based encryption. The presented work is also related to verifiable outsourced ABE. While the outsourced schemes [13, 29, 15] protect the data privacy, they provide no guarantee to the correctness of the transformation performed by the cloud server. Lai et al. implemented a verifiable outsourced ABE approach by attaching an additional encrypted random message to the real message and computing the digest of the two messages together [16]. The receiver recovers both messages and checks the digest to verify the correctness of the received messages. Similarly, Li et al. proposed an outsourced ABE with checkability by adding a redundant pre-shared k-length bit string to each message. The receiver can detect the dishonest action by checking the bit string. Although these schemes can verify the correctness of the transformation at the cloud server, they provide no guarantee to the completeness of the forwarded messages. That is, checking whether the cloud completely forwards all messages that a user is qualified to receive. This problem is somehow related to verifiable computation [11, 24] and verifiable searchable encryption [28], which guarantee the returned result is correct and complete. However, these schemes rely on either expensive fully homomorphic encryption or the pre-sharing of certain knowledge about the underlying message between the sender and the receiver, which is obviously infeasible in the privacy-preserving targeted broadcast applications.

7. CONCLUSION

In this paper, we present two multi-cloud outsourced-ABE schemes for privacy preserving targeted broadcast for IoT devices. By enforcing the collaboration between multiple clouds, our schemes significantly reduce the computational overhead at the resource-constrained IoT devices. The new schemes protect data privacy, attribute privacy and access policy privacy. To the best of our knowledge, this is the first work to utilize multi-cloud structure to prevent the disclosure of attributes and access policies in outsourcing. Our schemes also provide verifiability, which allows receivers to verify the correctness and completeness of the outsourced operations. Through intensive security analysis and experiments with simulated IoT devices and commercial cloud platforms, we demonstrate the security guarantees and outstanding performance of the proposed schemes.

8. ACKNOWLEDGMENT

This work was partially supported by the National Science Foundation under Award DGE1565570 and KU RIC Strategic Initiative Grant under Award INS0073037.

9. REFERENCES

[1] Java Pairing-Based Cryptography Library. http://gas.dia.unisa.it/projects/jpbc/.

[2] Raspberry Pi. https://www.raspberrypi.org/.

[3] The 10 most popular Internet of Things applications. http://iot-analytics.com/10-internet-of-things-applications/.

[4] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, IEEE Symposium on*, 2007.

[5] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[6] R. Bobba, O. Fatemieh, F. Khan, C. A. Gunter, and H. Khurana. Using attribute-based access control to enable attribute-based messaging. In *ACSAC*, 2006.

[7] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology*, 2005.

[8] P. Cerwall. Ericsson mobility report, 2015.

[9] Z. M. Fadlullah, N. Kato, R. Lu, X. Shen, and Y. Nozaki. Toward secure targeted broadcast in smart grid. *Communications Magazine, IEEE*, 2012.

[10] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology*, 1993.

[11] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology*. 2010.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, 2006.

[13] M. Green, S. Hohenberger, and B. Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security Symposium*, volume 2011, 2011.

[14] V. Gulisano, M. Almgren, and M. Papatriantafilou. Online and scalable data validation in advanced metering infrastructures. In *Innovative Smart Grid Technologies Conference Europe*. IEEE, 2014.

[15] J. Hur. Attribute-based secure data sharing with hidden policies in smart grid. *Parallel and Distributed Systems, IEEE Transactions on*, 2013.

[16] J. Lai, R. H. Deng, C. Guan, and J. Weng. Attribute-based encryption with verifiable outsourced decryption. *Information Forensics and Security, IEEE Transactions on*, 8(8):1343–1354, 2013.

[17] F. Li and B. Luo. Preserving data integrity for smart grid data aggregation. In *SmartGridComm*, 2012.

[18] F. Li, B. Luo, and P. Liu. Secure information aggregation for smart grids using homomorphic encryption. In *SmartGridComm*. IEEE, 2010.

[19] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang. Securely outsourcing attribute-based encryption with checkability. *Parallel and Distributed Systems, IEEE Transactions on*, 25(8):2201–2210, 2014.

[20] L. Liu, M. Esmalifalak, Q. Ding, V. A. Emesih, and Z. Han. Detecting false data injection attacks on power grid by sparse optimization. *Smart Grid, IEEE Transactions on*, 5(2):612–621, 2014.

[21] R. Lu, X. Liang, X. Li, X. Lin, and X. S. Shen. Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. *Parallel and Distributed Systems, IEEE Transactions on*, 2012.

[22] R. Lu, X. Lin, and X. Shen. Spoc: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency. *Parallel and Distributed Systems, IEEE Transactions on*, 2013.

[23] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 2012.

[24] C. Papamanthou, R. Tamassia, and N. Triandopoulos. Optimal verification of operations on dynamic sets. In *Advances in Cryptology*. Springer, 2011.

[25] Y. Ren, R. W. N. Pazzi, and A. Boukerche. Monitoring patients via a secure and mobile healthcare system. *Wireless Communications, IEEE*, 2010.

[26] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[27] L. Yang and F. Li. Detecting false data injection in smart grid in-network aggregation. In *Smart Grid Communications, IEEE Conference on*. IEEE, 2013.

[28] Q. Zheng, S. Xu, and G. Ateniese. Vabks: verifiable attribute-based keyword search over outsourced encrypted data. In *Infocom, IEEE*, 2014.

[29] Z. Zhou and D. Huang. Efficient and secure data storage operations for mobile cloud computing. In *Network and Service Management*, 2012.