REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
	Technical Report	-

4. TITLE AND SUBTITLE
Secure and Turstworthy Task Offloading in Mobile Cloud

5a. CONTRACT NUMBER
W911NF-11-1-0191

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS
Bo Li, Dijiang Huang, Weiping Peng, Bing Li

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
Arizona State University
ORSPA
P.O. Box 876011
Tempe, AZ 85287 -6011

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
56078-CS.9

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
As a Promising combination of mobile technology and cloud computing, mobile cloud has been analyzed and treated as an effective way to off-load some of the heavy workloads from mobile devices. The most recent progress in this area is to construct a small ad-hoc cloud environment located close to the mobile user. The key benefit for such design is that it's applicable even when no internet connection is available. Existing offloading algorithms for mobile cloud have taken comprehensive considerations on computation capacities, communication latency, and even available battery life on mobile devices. However, in mobile cloud environment, the performance related

15. SUBJECT TERMS
Security, trust, mobile cloud

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dijiang Huang
UU	UU	UU	UU		19b. TELEPHONE NUMBER 480-965-2776

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

REPORT DOCUMENTATION PAGE (SF298)
(Continuation Sheet)

Continuation for Block 13

Proposal/Report Number: 56078.9-CS

Report Title: Secure and Turstworthy Task Offloading in Mobile Cloud
Report Type: Technical Report

Secure and Trustworthy Task Offloading in Mobile Cloud

Bing Li, Bo Li, Dijiang Huang, and Weiping Peng

Abstract—As a promising combination of mobile technology and cloud computing, mobile cloud has been analyzed and treated as an effective way to off-load some of the heavy workloads from mobile devices. The most recent progress in this area is to construct a small ad-hoc cloud environment located close to the mobile user. The key benefit for such design is that it's applicable even when no Internet connection is available. Existing off-loading algorithms for mobile cloud have taken comprehensive considerations on computation capacities, communication latency, and even available battery life on mobile devices. However, in mobile cloud environment, the performance-related factors are not sufficient to characterize the dominant concerns in off-loading. In this paper, we propose a new approach to establish trust in mobile cloud. It creates temporary groups without revealing group member information to any network participant. Compared to traditional group key management schemes, this method provides anonymity protection for group members based on Attribute Based Encryption(ABE) algorithm. Furthermore, we develop a novel task off-loading decision algorithm with trust factors, which, according to our knowledge, is the first of such a kind. With the proposed approach, the trust relationship between mobile devices is incorporated into off-loading decisions, making it more practical in real-world application scenarios.

Index Terms—mobile cloud, secure, trust, off-loading

1 Introduction

Traditional mobile cloud computing refers to mobile devices running cloud-based applications or consuming services provided from remote cloud system. This architecture is of great importance as it provides an effective approach to off-load heavy tasks from mobile devices to cloud systems. Instead of executing these tasks locally, the device sends task-related contexts to remote cloud, which essentially provides the mobile device with powerful computation capabilities. Some applications, for instance Matlab, that were computationally impossible to be executed on mobile devices now run smoothly in their cloud versions.

Although it boosts the mobile devices with more powerful capability, such a service architecture is heavily reliant on the assumption that the network connections between mobile devices and remote cloud services are fast and reliable. With the rapid technology development in wireless communications, the latest 4G system is much faster with better QoS than previous generations. However, such high speed and high reliability are built on a well constructed and maintained network infrastructure with a broad access coverage. There are many scenarios where such wireless connections are not available. One of them is the mountainous area. Due to the low density of population and the complex terrain landscape, it is economically impossible for telecommunication providers to seamlessly cover these areas. Thus, when people are traveling on a road trip in the area, it is normally difficult to get real-time cloud service through mobile devices. Another typical scenario of this kind is disaster rescue. Be it an earthquake, a tsunami, or landslides, the rescue tasks normally involve large amount of data processing. Traditional telecommunication infrastructures can hardly be recovered within the first few days, which is critical for life-saving tasks. Under such situations, a mobile cloud system formed by a loose set of heterogeneous mobile devices is desirable.

In this mobile cloud, a task can be off-loaded from a user's mobile device to nearby devices so that more computation power is involved in executing the task. Unlike traditional cloud systems, where hardware devices are relatively static and well administrated by the same service provider, devices in a mobile cloud normally belong to different entities with a dynamic network topology. Therefore, from a user's perspective, the cloud service providers are not guaranteed to be always available, which makes it quite challenging for off-loading decisions.

As an important part of mobile cloud, existing task off-loading algorithms [1, 2, 3] make the off-loading decisions based on hardware computation capability, real-time connection quality, and current local system status. These algorithms work well in traditional cloud computing scenario. However, as mentioned above, mobile cloud consists of heterogeneous devices, whose identity and origin are hard to be monitored. The ad-hoc nature makes it important to establish trust among mobile devices. Once such trust is set up, off-loading decisions can be made based on both performance and trust.

- *B. Li, and D. Huang, are with the School of Computing Informatics and Decisions Systems Engineering, Arizona State University, Tempe, AZ 85281. E-mail: { Bing.Li.4, Dijiang.Huang}@asu.edu*
- *B. Li is with School of Information Science and Engineering,Yunnan University, Kunming 650091, China. E-mail: boli18@asu.edu*
- *W. Peng is with School of Computer Science, Henan Polytechnic University, Jiaozuo 454000, China. E-mail: wpeng12@asu.edu*

Envisioned with such a need in mobile cloud, we propose a new approach to set up trustworthy working groups in mobile cloud environment. Traditional group management schemes rely on a set of key-encryption-keys (KEKs) and a hierarchical key management structure to reduce the complexity. Usually, a trusted server is responsible for group management by maintaining the minimal set of KEKs that are shared by legitimate group members. During the communication, a group session key is encrypted using KEKs and then it is distributed to group members for decrypting secret messages. This approach works well in a centralized group management scenario, i.e., a centralized key server is responsible for managing the group formation. However, in most mobile cloud scenarios, such a dedicated and trusted key server does not exist. It is usually desirable that each group member can self-manage the group/subgroup formation without relying on an online centralized group management server.

To solve such issue, we target to use ABE algorithms instead of traditional public key algorithms. The key benefit of ABE is that it reduces the workload for key management. A user who needs to establish a secure group does not need to know all the keys for each individual group member. Instead, he only needs to specify a certain policy, which identifies the attributes needed to decrypt a message. This is quite suitable in distributed environment. However, in traditional ABE algorithms, the policy content is transmitted in plaintext. This implies another critical concern with existing group management: membership anonymity. Information concerning the group formation, such as group member identities and group size, are usually the targets for attackers.

To address the above issues, we propose a new trust group management scheme based on ABE for mobile cloud. Our approach achieves better performance than previous solutions by using a novel gradual exposure solution named *Efficient and Anonymous Group Setup Scheme* (EA-GSS). Instead of requiring every message receiver to use each of their attributes to decrypt the message, a receiver is able to discover his eligibility through a guided process by discovering the eligibility with the maximum $O(h)$ steps, where h is the height of the attribute policy tree. In addition, unlike most of previous solutions, the attribute management of EA-GSS is decentralized, so that any node can be the group controller or one of the group members in different group setups. The update processes for both group session keys and attributes can be distributed among network entities.

With such a scheme, we further propose an off-loading decision algorithm which considers both performance factors and trust aspects. As mentioned before, devices in mobile cloud cannot be trusted equally due to their diverse backgrounds. When making off-loading decisions, the service consumer should not only consider the task completion time but also whether it is appropriate to off-load a sensitive task to a strange device. To this end, we differentiate the sensitivity of different tasks by assigning a minimum trust threshold for each of them. When applying task off-loading decisions, a task is off-loaded only when the service consumer has a higher trust on the service provider than the task's trust threshold.

In summary, our proposed scheme achieves the following contributions:

- It applies an attribute-based trust group establishment mechanism that incorporates the policy-based data access control in ciphertext. Only eligible group members can discover the group formations;
- It provides privacy protection on group access policies in that any ineligible group node cannot reveal the information of the group policy even if they collude together;
- It incorporates trust with performance into consideration when making off-loading decisions. A thorough performance evaluation for the proposed trust-incorporated off-loading algorithm is conducted.

The remainder of this paper is organized as follows: Section 2 provides the related work on secure group management, ABE-based solutions, and task off-loading algorithms; Section 3 presents the system models and preliminaries of an ABE scheme; detailed description for the proposed group management scheme is provided in Section 4; in Section 5, we propose a novel off-loading algorithm which covers both performance factors and trust; the performance and security analysis of the proposed schemes is presented in Section 6; we conclude this paper in Section 7.

2 Related Work

In this section, we review the related work in both trust group management and task off-loading algorithms in cloud environment. We analyze the pros and cons for each work and identify the features and functionalities we need to achieve with our proposed solution.

2.1 Group Management

Traditional solutions for group setup are based on a key distribution center (KDC) and a hierarchical tree [4]. Any node needs to keep a record of other users' identities before it establishes a group. When a node sets up a group, it works with the KDC to resolve the minimum set of keys shared among all the group members. Then this node sends out the group session key encrypted with each of the keys. In this way, the KDC who possesses knowledge of the key hierarchy is heavily involved in the setup of every group.

In 2007, J. Bethencourt *et al*. proposed a new ABE scheme called Ciphertext-Policy ABE (CP-ABE) [5]. In this scheme, each node is assigned with a set of attributes, and each attribute corresponds to a private key. The policy is defined in the encryption process. Only eligible entities with satisfactory attributes can successfully

decrypt the ciphertext. The policy is sent out in plain text together with the ciphertext, so that a receiver can check if it satisfies the policy before decryption. CP-ABE makes it possible to apply access control to individual messages. As such, a sender can specify all the required attributes without knowing the keys of each individual receiver in advance. Moreover, this scheme is secure against collusion attacks.

In CP-ABE, the policy is transmitted in clear text. Any receiver of the ciphertext is able to check the policy even if his attributes do not satisfy. Thus, the entire policy is exposed to attackers, causing a serious privacy issue. To hide the policy without interrupting the decryption process, several schemes [6, 7] are proposed to make sure that a message receiver gets no information about the policy if he fails in decryption. All these hidden-policy schemes share a common problem that a node receiving the ciphertext cannot determine if it is an eligible receiver before it completes the entire decryption process. This incurs heavy computation overhead in group setups for each node who receives the group establishing message. When a node sends out a session key under a policy in broadcast manner, any receiver of the message will have to decrypt it so as to determine if it is part of the group. Since the number of group members is usually much smaller than the total amount of nodes in network, the additional computation cost for non-group members is not negligible. Furthermore, the scheme in [6] is not designed in a public-key pattern. Only the group controller has the ability to set up a group, which greatly limits its application scenarios.

To reduce the computation overhead, D. Huang *et al*. proposed to gradually expose the policy in multiple steps [8]. At each step, only one attribute is exposed. In this way, a receiver can decrypt a message with one attribute at each step. Once the current decryption step fails, the receiver immediately knows he is not eligible. This approach greatly reduces the computational cost because ineligible nodes do not have to complete the entire decryption process. However, it only supports AND-gates in the attribute policy which limits the flexibility of the policy. Moreover, it exposes one more attribute to the receiver if he fails in the decryption process. Comparatively, the proposed trust group scheme in this paper enables the use of OR-gates, and protects the privacy of the attribute policy so that ineligible nodes cannot get any information on any attribute in the policy that they do not have.

2.2 Off-loading Decision

Off-loading decision problem has been studied by many researchers for a variety of application scenarios. For traditional mobile cloud computing environments, there exists a infrastructure-based network which allows mobile devices to off-load the application tasks to a remote cloud service. Several existing works such as MAUI [1], COMET [2], and ThinkAir [3] are proposed for such application scenarios. In MAUI, the goal of the off-loading decision is to achieve best energy savings for the mobile device under the constraint of the real-time connectivity. It supports Microsoft .NET Common Language Runtime (CLR) environment for application partitioning, which provides a fine-grained code off-load for mobile devices. The system constructs profiles for devices, programs, and network connections. The decision problem is modeled as an integer linear programing (ILP) problem based on the profiles. Experiment results shown that MAUI system achieves a good savings in energy consumption under wifi environment with small RTT values. But the energy consumption is larger than executing the tasks locally when using 3G connections with 220ms RTT for video game and chess game applications. In COMET, the authors proposed a runtime system which enables multi-threaded applications to migrate among multiple machines. It runs on top of Java VM specifications and handles multi-threaded environments on Android systems. The evaluation results showed that for most of the applications tested, the wifi off-loading significantly reduced the absolute execution time while the 3G off-loading does not guarantee to consume less time than local executions. ThinkAir provides a off-loading capability from method level. Each method may be executed on different VMs in the cloud. Off-loading decisions are made based on the profiles constructed for hardware, software, and network. The performance comparison in terms of execution time and energy consumption showed that wifi off-loading performs better than 3G off-loading and local execution has the worst performance in both criteria.

All the above-mentioned solutions are based on the assumption that a static network architecture with cloud service is available for task off-loading. In mobile cloud environment, such an assumption is not always true. Especially, when the nodes requiring offloading services and those providing such services are moving simultaneously following different path, such a problem gets much more complicated. For such scenarios, Scavenger[9], Serendipity[10] and CirrusCloud[11] are some typical off-loading solutions from mobile device to mobile device. An off-loading system normally makes profiles for devices and jobs. For device profiling, it records its computation power and energy consumption; for job profiling, it logs its component composition, the computation requirement, and the I/O data size. With these two types of profiles, the system is able to quantify the dynamics of the entire network and make decisions using jobs as basic units. In Serendipity[10], which is a framework to enable the offloading of applications from smartphones to other mobile devices, jobs were represented graphically as directed acyclic graphs(DAGs) of PNP-blocks, and three task allocation algorithms in the context of three models with different contact knowledge and control channel availability assumptions were put forward to minimize the completion time. While allocating tasks, only online algorithms were considered

to minimize the completion time. Batch scheduling algorithms were not taken into account. Different from Serendipity, which only offload applications to other mobile devices, CirrusCloud[11] can enable mobile devices to offload applications to whatever resources they can access, ranging from remote central data center to local cloudlets and to nearby mobile devices. The authors present an robust heuristic that can tolerate intermittent connectivity and identify a number of issues that should be investigated, such as how to provide continuous execution with intermittent connectivity and how to optimally use the mixture of available resources.

A flexible, seamless, and lightweight method for mobile device off-loading has been studied thoroughly [12, 13, 3]. Application partitioning and component scheduling to cloud resources is another important problem for academia researchers [14, 15, 16]. Dynamic mapping (matching and scheduling) algorithms for a class of independent tasks in heterogeneous distributed computing systems were studied in [17, 18]. In the study, a collection of typical heuristics, ranging from online scheduling heuristics, batch scheduling heuristics, to metaheuristics, have been selected, adapted, implemented and examined for heterogeneous computing environments. The comparisons of the heuristics provided in [17, 18] have been regarded as a starting point for choosing heuristics to apply in different scenarios and for selecting heuristics against which to compare new, developing techniques.

3 Models and Preliminaries

In this section, we present the basic system models and security-related models. A general description of ABE scheme is also provided.

3.1 Network Model

In our scheme, each node in network is a mobile device. There are two goals for the proposed scheme. The first one is to identify a set of trusted nodes for each device so that it can off-load sensitive tasks based on its trust on each node. For this purpose, each node in network is assigned with a unique identifier (UID) and a set of attributes. UID itself can be treated as one attribute. A trusted third party (TTP) is in charge of setting up global parameters for the entire network. A network node that defines and manages one or more attributes is named as the authority of the attribute(s). Each attribute is defined and managed by the same authority. Any network node can work as an authority. When a node needs to set up a group (we denote this node as the *Group Founder* of the group), it generates an attribute policy using logic gates to connect attributes. Therefore, unlike traditional group management schemes, the *Group Founder* does not need to know each of the group members' keys. All it concerns is the attribute policy that a group member needs to satisfy. For example, if a university principal wants to set up a video conference with deans from every department, he does not need to know each individual key of the deans. Instead, he can set up this group using the policy: $\{Principal\}\,OR\,\{Dean\}$. Any party in the network that possesses neither $\{Principal\}$ nor $\{Dean\}$ will not be able to decrypt any secret information in the group.

To complete a secure group setup, the *Group Founder* needs to send out a group session key to all the other members using the EA-GSS scheme with the attribute policy it created. Note that unlike [6], any node can be a *Group Founder* and set up a group. There is no additional requirements between a *Group Founder* and a group member.

In secure group communications, it is necessary for the *Group Founder* to authenticate itself to other members. The goal is to prove that the founder itself satisfies the attribute policy. Otherwise, an attacker can easily set up a group with network nodes that he is not supposed to communicate with. In our design, this requirement is met by sharing information of each attribute secretly among attribute holders only. A node without such information cannot use the attribute for encryption. The details are illustrated in **Algorithm 4**.

The second goal of the proposed scheme is to make an off-loading decision based on trust and performance. For performance consideration, we assume the hardware capacity and the network connectivity for each node is known to the network. Therefore, similar to the related works, off-loading decisions can be made based on profiles of device and network for performance concerns. To model the dynamic movement of the mobile nodes, we follow the work in [19] and assume the state of the entire network is stable within a short period of time when a decision is made. In other words, we slice the time domain into short time-frames so that in each frame, the network connectivity relationship is relatively stable. Every time when such a decision is needed, a connected graph *G=(V,E)* is generated to represent the current state of the cloud. The vertices, *V*, represent the mobile nodes while the edges, *E*, represent the connection links between nodes. For each pair of nodes, there is at most one direct link between them. For nodes that are not directly connected, they may be able to set up an end-to-end connection based on the underlying routing protocols [20].

The bandwidth between each node is critical in estimating the communication time cost for an off-loading job. Many methods have been proposed to model the bandwidth for IEEE 802.11 based ad hot networks[21]. In the rest of this paper, we follow a popular estimation method to model the end-to-end bandwidth with respect to the number of hops (*HopNumber*) and the bottleneck bandwidth (*MinBandwidth*) in the routes as follows(see [22, 23] for more details):

if (*HopNumber* == 1) Bandwidth = MinBandwidth
else if (*HopNumber* == *2*) Bandwidth = MinBandwidth/2
else if (*HopNumber* == *3*) Bandwidth = MinBandwidth/3

else Bandwidth = MinBandwidth/4.

This equation offers the upper bound of the available end-to-end bandwidth via multi-hop routes. In Section 6, the nodes are assumed to be connected by 802.11b networks with ad hoc model, and the *MinBandwidth* is set to be $11Mbps$, or equally $1.375MB/s$.

3.2 Task and Notation

Each task in our design is modeled as a batch of independent jobs. A job is processed once it is generated. Depending on its context, e.g. available mobile nodes, trust level, available local resources, etc., a job may be off-loaded or executed locally.

In the rest of this paper, we refer to a mobile device/user who's making an off-loading decision as an **Assigner**, and the mobile device who provides such mobile cloud service to the assigner as an **Assignee**. We do not further differentiate a mobile device from a user as a user can only conduct actions through his mobile device.

3.3 Attack Model

In this paper, we assume attackers are trying to compromise the scheme for two goals: (1) joining in a trust group without eligible attributes; (2) retrieving the group formation information. To achieve the first goal, an attacker needs to retrieve the session key of the target group. This can be done by exploiting vulnerabilities within the secret protection functionality of the group key distribution scheme. If this is difficult, the attacker can change to target at the group membership information. The identity information of the authorized key receivers is stored and protected within the key distribution message. The attacker needs to analyze this message and the protection technique of the identity information so as to expose this information or gain certain amount of knowledge about it. The consequence of a successful attack is to gain unauthorized information, identify valuable nodes and launch further attacks on them.

Either goal mentioned above involve breaking into the cryptographic schemes. Now we define the security of the proposed scheme based on a game between a challenger and an attacker.

We use the symbol S_A to represent the set of all the attribute authorities. A certain number of authorities are corrupted by the attacker, the set of which is denoted as S_{cA}. We also use U_A to denote the universe of all the attributes.

3.4 Preliminaries of CP-ABE

CP-ABE is based on bilinear pairing computation. Suppose G_0 is an additive group and G_1 is a multiplicative group. Both of them are with a large prime order p. Discrete Logarithm Problem (DLP) is hard in both G_0 and G_1. Define a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. This map has three properties:

- Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$, for any $P, Q \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$;
- Nondegeneracy: $e(g, g) \neq 1$, g is the generator of $\mathbb{G}_0$;
- Efficiency: Pairing can be efficiently computed.

In CP-ABE, there are three types of keys: master key, public key and private key. A TTP is required to generate a set of public parameters and the master key. The TTP is not involved in network communication. It can be offline. There are four basic algorithms in CP-ABE: **Setup**, **Encrypt**, **KeyGen** and **Decrypt**. In **Setup**, the TTP chooses two random exponents $\alpha, \beta \in Z_p$. A public key is formatted as $< G_0, g, h, f, e(g, g)^{\alpha} >$ while the master key is (β, g^{α}). Here $h = g^{\beta}$, $f = g^{\frac{1}{\beta}}$. The public key is published by the TTP. When a party wants to encrypt a message M, it runs the **Encrypt** algorithm, which takes the public key, the message M and a policy tree T as inputs. The **KeyGen** algorithm is used to generate private keys based on the master key and a set of attributes. For each network node, the TTP runs the **KeyGen** once to generate a private key according to the node's attributes. When a node receives the ciphertext, it runs the **Decrypt** algorithm to get the encrypted data.

3.5 Security Model

The security of the proposed scheme can be modeled in terms of a game between a challenger and an adversary. The challenger simulates the trusted third party (TTP) and attribute authorities while the adversary tries to impersonate as a number of normal network nodes. The game consists of the following five steps:

Setup. The challenger runs the **GlobalSetup** algorithm and feeds back to the adversary the global parameters.

Phase 1. The adversary asks for a number of user keys from the challenger. The amount of keys and users are arbitrary. The challenger runs the **NodeJoin** algorithm for each user involved in the requests and returns the corresponding secret information back to the adversary. The adversary can act as these users to request for attributes from the challenger. The challenger then runs the **AuthoritySetup** algorithm to create parameters for attribute authorities and run the **KeyGen** algorithm on behalf of the authorities and the TTP. In other words, **KeyGen** in this game is conducted by the challenger itself. It creates new authorities only when needed.

Challenge. The adversary provides two messages M_0 and M_1 to the challenger together with an access policy A. A satisfies that none of the users created by the challenger has attributes satisfying A. The challenger flips a coin b and encrypts M_b using A and returns the ciphertext C back to the adversary.

Phase 2. The adversary can ask for more attributes and users from the challenger. But if any user can gain satisfactory attribute combinations for A, the challenger aborts the game. The adversary can always request for any public attribute key.

Table 1
Notations

Terms	Meaning
$\mathbb{G}_0$	a bilinear group with a prime order p
$\mathbb{G}_1$	a multiplicative group with the same prime order p
$e(\cdot)$	a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$
$ROOT$	a global constant value $ROOT \in \mathbb{G}_1$ as ide--ntification of the secret message protected with the policy
$Enc_k(\cdot)$ $Dec_k(\cdot)$	a symmetric encryption algorithm $Enc_k(\cdot)$ and the corresponding decryption algorithm $Dec_k(\cdot)$ in $\mathbb{G}_1$
encryption sequence	a conjunctive clause representing the seque--nce of attributes in the encryption process
decryption sequence	a conjunctive clause representing the seque--nce of attributes in the decryption process
A_i or A_n	an attribute, A_i is used for denoting an ind--ividual attribute, A_n denotes the n-th attri--bute in a sequence
A_{Pub}	a public attribute shared among all the net--work nodes, the corresponding values sto--red at each node are (I_{Pub}, T_{Pub}), $I_{Pub} \in \mathbb{Z}_p, T_{Pub} \in \mathbb{G}_0$

Guess. The adversary outputs a guess b′ of b.

The adversary's advantage in this game can be defined as $ADV = P[b' = b] - \frac{1}{2}$. The proposed scheme is secure if for all the polynomial time adversaries, the advantage is at most negligible in the game.

4 ABE FOR COMMUNICATION GROUP SETUP

In this section, we present the EA-GSS scheme for secure communication group setup, which is developed based on ABE algorithms [5, 8]. We also illustrate the management of group keys and attribute keys. Before introducing details of the scheme, we give a summary of notations in Table 1.

4.1 EA-GSS

Attributes of a node can be any value in strings. In CP-ABE, these values are converted into mathematical values using hash functions. In EA-GSS, each attribute string A_i corresponds to a triplet (T_i, I_i, k_i). Mapping from a string to such a triplet is not defined by hash functions but determined by the authority of A_i. An access policy can be represented in Disjunctive Normal Form (DNF) of attributes. In each conjunctive clause of the DNF, the sequence of attributes is enforced by the encryptor. We name the sequence of attributes when encrypting a clause as encryption sequence and the opposite sequence as decryption sequence. We define a public attribute A_{Pub} in EA-GSS. Unlike other attributes, A_{Pub} is associated with an ordered pair (T_{Pub}, I_{Pub}). For each conjunctive clause, the encryptor adds A_{Pub} at the end of the encryption sequence.

In EA-GSS scheme, a **GlobalSetup** algorithm is run by a TTP to generate global parameters for the system. The parameters are assigned to each node before it joins in the network. For each node joining in the network, the TTP runs a **NodeJoin** algorithm to generate a unique secret for the node. The input of **NodeJoin** is the node's UID while the outputs are {D_{UID}, $X_{Pub,UID}$, Y_{Pub}, $Z_{Pub,UID}$}. For each attribute, the attribute authority runs an **AuthoritySetup** algorithm to generate secrets associated with the attribute. This can be done after the node is in the network. Additionally, there are other three basic algorithms in EA-GSS: **KeyGen**, **Encrypt** and **Decrypt**, which are illustrated with details in the following.

The **GlobalSetup** algorithm and **NodeJoin** algorithm are defined as in **Algorithm 1** and **Algorithm 2**.

Algorithm 1 GlobalSetup

1: Choose a bilinear group $\mathbb{G}_0$ with a large prime order p. g is the generator of $\mathbb{G}_0$;
2: Choose two random values $\alpha, \beta \in \mathbb{Z}_p$;
3: Publicly define a global constant value $ROOT \in \mathbb{G}_1$ as identification of the secret message;
4: Publicly choose a symmetric encryption algorithm $Enc_k(\cdot)$ and the corresponding decryption algorithm $Dec_k(\cdot)$ in $\mathbb{G}_1$;
5: Define and publish a public attribute shared among the network nodes, $(S_{Pub}, T_{Pub}), S_{Pub} \in \mathbb{Z}_p, T_{Pub} \in \mathbb{G}_0$;
6: The global parameters are {$\mathbb{G}_0$, g, g^β, $e(g,g)^\alpha$, $Enc_k(\cdot)$, $Dec_k(\cdot)$, (S_{Pub}, T_{Pub}), $ROOT$}, global secrets are {β, g^α}.

Algorithm 2 NodeJoin

1: For each node with UID joining in the network, generate a random number $r_{UID} \in \mathbb{Z}_p$ and store it securely;
2: Calculate and assign $D_{UID} = g^{(\alpha + r_{UID})/\beta}$ to the node;
3: Calculate and assign to the node:

$$X_{Pub,UID} = g^{r_{UID}} T_{Pub}^{r_{Pub}},$$

$$Y_{Pub} = g^{r_{Pub}},$$

$$Z_{Pub,UID} = e(g,g)^{r_{UID} I_{Pub}}.$$

where $r_{Pub} \in \mathbb{Z}_p$ is a random number for each node;
4: Assign to the node {D_{UID}, $X_{Pub,UID}$, Y_{Pub}, $Z_{Pub,UID}$}.

Each authority that manages an attribute A_i will have to run the **AuthoritySetup** algorithm (**Algorithm 3**) to set up the secrets associated with A_i.

Algorithm 3 AuthoritySetup

1: For each attribute A_i, choose two random numbers $I_i, k_i \in \mathbb{Z}_p$;
2: For each attribute A_i, choose one random value $T_i \in \mathbb{G}_0$.

The **KeyGen** algorithm generates the private keys corresponding to each attribute for each node holding this attribute. It is a cooperative algorithm between an authority and the TTP, which is defined in **Algorithm 4**.

The **Encrypt** algorithm works like this: following the encryption sequence of each conjunctive clause, denote

Algorithm 4 KeyGen

1: For each attribute A_i assigned to the node with UID, the authority passes UID, I_i and T_i to the TTP;
2: The TTP computes and sends back to the authority:

$$X_{i,UID} = g^{r_{UID}} T_i^{r_i},$$
$$Y_i = g^{r_i},$$
$$Z_{i,UID} = e(g,g)^{r_{UID} I_i}.$$

where $r_i \in \mathbb{Z}_p$ is a random number;
3: The authority assigns $X_{i,UID}$, Y_i and $Z_{i,UID}$ to the node together with T_i, I_i and k_i generated from Algorithm 3.

each attribute from I_1 to I_m, where m is the number of attributes in the clause. Choose a random value $s \in Z_p$ and set $I_0 = s$. Given such a clause, the encryption process on group session key S_k is defined in **Algorithm 5**. A complete encryption includes such a process for every clause but the overlapping parts between clauses. For example, given a policy {A **AND** B **AND** C} **OR** {A **AND** B **AND** D}, where A, B, C, and D are four attributes, the simplified form is {A **AND** B} **AND** {C **OR** D}. The encryptor calculates the ciphertext for {A **AND** B **AND** C} first and then use the results for {A **AND** B} to finish {A **AND** B **AND** D} = {A **AND** B} **AND** {D}.

Algorithm 5 Encrypt

1: Calculate $C = S_k e(g,g)^{\alpha s}$, $C' = g^{\beta s}$ and $C'' = Enc_{S_k}(ROOT)$, where $s \in Z_p$ is a random value, S_k is the message to be encrypted;
2: Start from the beginning of the clause in the encryption sequence;
3: For each attribute A_n, if a triplet $(C_{1,n}, C_{2,n}, C_{3,n})$ has already been calculated, move to the next attribute A_{n+1} and restart step 3 with A_{n+1}; else, goto step 4;
4: Choose a random number $t_n \in \mathbb{Z}_p$;
5: Calculate:

$$C_{1,n} = g^{(I_{n-1} - I_n) t_n},$$
$$C_{2,n} = T_n^{(I_{n-1} - I_n) t_n},$$
$$C_{3,n} = (k_n t_n)^{-1}.$$

$1 \le n \le m$;
6: Calculate $C_{1,m+1} = g^{(I_m - I_{Pub})}$, $C_{2,m+1} = T_{Pub}^{(I_m - S_{Pub})}$.

The **Decrypt** algorithm works in the decryption sequence. Note that the first attribute in decryption sequence is always A_{Pub}. The decryption process is defined in **Algorithm 6**.

When the **Decrypt** algorithm succeeds, S_k is the group session key embedded in C.

4.2 Group Setup

With the above scheme, any node that wants to set up a trust group can construct a group attribute policy. This is straightforward since there is no need for any knowledge about other group members' keys. It specifies the attribute policy that the group members need to satisfy. Then this node distributes a group session key to the group members using the established policy. In

Algorithm 6 Decrypt

1: Start from the public attribute A_{Pub};
2: For each attribute A_n that the decrypter possesses, compute:

$$\frac{Z_{n,UID_{dec}} \cdot e(X_{n,UID_{dec}}, (C_{1,n})^{k_n C_{3,n}})}{e(Y_n, (C_{2,n})^{k_n C_{3,n}})}$$
$$= e(g,g)^{r_{UID_{dec}}(I_{n-1})};$$

3: If $e(g,g)^{r_{UID_{dec}}(I_{n-1})}$ is one of the decrypter's private keys, then go to step 2 with attribute A_{n-1}; else go to step 4;
4: Calculate

$$S_k = C/(e(C', D)/e(g,g)^{r_{UID_{dec}}(I_{n-1})}.$$

If $Dec_{S_k}(C'') == ROOT$, Success; else, Failure.

EA-GSS, only the eligible members can correctly decrypt this key. Any session key updates can be performed by sending updated session keys to the group members according to the attribute policies.

The *Group Founder* authentication is incorporated in the EA-GSS scheme. This is because in **Encrypt** algorithm, only those who possesses the correct value of T_n, I_n and k_n has the ability to generate the correct ciphertext $C_{1,n}$, $C_{2,n}$ and $C_{3,n}$. Since T_n, I_n and k_n are exclusively shared as a secret among the nodes who are assigned with attribute A_n, only these nodes are able to generate the ciphertexts correctly.

4.3 Attribute Key Update

In addition to group key management, it is necessary to incorporate an attribute key update function in case a new node is assigned with an attribute after the network setup or a node is deprived of an attribute. The authority runs the **KeyUpdate** algorithm, which is defined in **Algorithm 7**, to conduct attribute key update.

Algorithm 7 KeyUpdate

1: For attribute A_i, choose two random values $I_i', \Delta k_i \in \mathbb{Z}_p$;
2: Encrypt I_i' and Δk_i for each eligible node UID that owns attribute A_i using the node's UID as the policy;
3: Each node updates its keys as $Z'_{i,UID} = (Z_{i,UID})^{I_i'/I_i}$, $k_i' = k_i + \Delta k_i$.

5 Off-loading tasks with trust

With the proposed group management scheme, we further investigate how to incorporate trust into off-loading decision making process. In this section, we firstly model the trust in mobile cloud environment. Based on such model, we further extend the work on off-loading decision algorithms.

5.1 Modeling Trust in Mobile Cloud

Based on the proposed group management scheme, each mobile device/user is able to assign an initial trust value to each group it involves in. This is treated as part of the persistent trust as surveyed in [24].

In this paper, we propose a new definition of trust based on the attributes of an trusted group. As mentioned in the previous sections, a policy by the group founder specifies a secure group. In other words, the trust to a group can be specified by the attribute combinations. In our previous work [25], we explored the idea of using set theory to model the anonymity exposure process for an ABE-based algorithm. If the size of the set corresponding to a certain policy attribute is small, then exposing such an attribute would leak more information than an attribute with a larger set. Similarly, if we apply such idea in trust modeling by substituting the set of an attribute with the set of a policy rule, finding a device satisfying such a rule for off-loading seems a good choice as that device is chosen from a narrowed-down set of candidates. However, such consideration is not always sufficient. For example, if we have a policy $A = \{Group1 \in CSE548 \wedge ASU\}$, which corresponds to the students in Group 1 of the course CSE 548 at ASU, and $|A| = 5$. Another policy is $B = \{Dean \wedge ASU\}$, which means the deans at ASU. Based on the number of schools at ASU, the size of B is more than 11. Thus, following the set-size based idea, B should be less trustworthy than A, which, in most cases, is not true.

To solve such issue, we adopt the idea from [26] by enforcing attribute ranking levels for the same attribute category. As in the above example, we can define $Group1 \in CSE548 < Dean$. With such definition, we further apply evidence theory [27] onto this case by defining Belief and Plausibility. Belief is defined as the level of trust a policy rule must have: $Bel(A) = \max_{a \in A} Trust(a)$. Plausibility is defined as the possible trust level that can be relaxed to regarding a policy rule: $Pl(A) = \min_{a \in A} Trust(a)$. With such definition, the trust for a certain rule should be a value between $Bel(A)$ and $Pl(A)$. The concrete value is dependent on the dynamic trust changes. In the above example, the trust for policy A and B can be interpreted as $Trust(ASU) \leq Trust(A) \leq Trust(Group1 \in CSE548)$ and $Trust(ASU) \leq Trust(B) \leq Trust(Dean)$ respectively. This implies under the same dyamic trust movement, we have $Trust(A) \leq Trust(B)$.

For dynamic trust, as it is based on the network state and context, we give users the capability to model such changes in terms of trust value. In other words, users can modify their trust values on each mobile device dynamically during the communication process after the trust groups are established. Specific approaches on how to model such trust changes are out of the scope of this paper. Interested readers may refer to related works [28, 29] in this area.

In the rest of this paper, we introduce two trust values: **device-trust** and **job-trust**. Device-trust refers to the trust value a mobile node/user has on another mobile devices. As mentioned before, this value is generated from both persistent trust and dynamic trust. It subjects to changes according to the context. For the same mobile device, different users may maintain different trust values due to their different context. For the same user, his trust on different devices may also be divergent.

Compared to device-trust, job-trust is used to define the trust requirement for each job. Divergence in job-trust reflects the difference in sensitivity of each job. For instance, in the same task, an authentication job requires much more trust on the executing party/environment than an ordinary mathematical computation job, though they may have a similar cost on computation power. Such differences are rooted from the different access policies of the data flows each job handles. In our design, each job in a task is assigned with a minimum trust threshold ($TRUST\_MIN$) and it is off-loaded to a mobile device only if the device has a higher trust value than the threshold. If such a mobile device is not found, either it is because no such a trust-worthy device exists or the job is too sensitive, the job will be executed locally on the assigner.

In addition to the minimum threshold, we further incorporate a penalty curve for each job. The curve is modeled as: $P(t_r), TRUST\_MIN \leq t_r \leq TRUST\_MAX$, where $TRUST\_MAX$ is the maximum trust value defined for the specific job, t_r is the trust of a certain resource r. It represents that if a job is off-loaded to a device with a trust value higher than $TRUST\_MAX$, no additional benefits would be achieved. Normally, a high trust value is achieved at the assigner itself unless a highly trust-worthy assignee exists. The penalty curve represents the cost an assigner may take for off-loading a job to a less trustworthy party. Thus, it, by nature, is a monotonically decreasing function. The concrete definition for the penalty curve of each job is managed by the assigner.

5.2 Trust-based off-loading decisions

In this subsection, we introduce our design on incorporating trust into the off-loading decisions. As mentioned in Section 2, all the related work on task off-loading algorithms are focused on the performance considerations. In our design, we not only consider the performance factors, but also the trust relations of jobs and mobile devices. Traditional off-loading algorithms can be categorized into online scheduling and batch scheduling. Online scheduling refers to the case when a job is scheduled (off-loading or local execution) immediately once it is ready. For batch scheduling, the assigner waits for a pre-defined time interval to accumulate a certain amount of jobs and then schedule they in a batch. This mode is quite useful in centralized off-loading architecture. In mobile cloud scenario, we adopt the idea from online scheduling algorithms to present our solution.

For each task in the off-loading scenario, we model three key features: trust, computation workload, and communication workload. For trust, the goal is that the chosen device for a certain job should have a trust value greater than or equal to the $TRUST\_MIN$ value of that job. If no such device exists in the mobile cloud, the job will be executed locally instead.

To make an off-loading decision, the first step is to assign a job-trust value to each job of the current task. With such a value, for each resource with a trust value T_r in the cloud, calculate the penalty $P(T_r)$ as the trust benefit an assigner can achieve from using this resource as the assignee.

After this, the assigner calculates the computation cost and communication cost for off-loading the job to each available resource. To evaluate the performance aspects, several different metrics are used in previous works. Minimum execution time (MET) is used to find the mobile resource which takes the least time to execute the target job. This criteria is effective if the jobs are mostly time-consuming. In other words, if the communication cost can be ignored, then we can make the off-loading decisions based on MET. However, in our targeted network model, communications are not as reliable as in the infrastructure-based cloud. Nodal mobility makes connections between two nodes on and off more frequently. To this end, we choose to user minimum completion time (MCT) as the metric for performance considerations. The benefits for using MCT in mobile cloud is that it considers both the communication time delay and the computation delay. In other words, we model the time consumption for executing a job. Namely, the total offloading time t_{ij} for job t_i on resource s_j is the sum of the computation time t_{ij}^{comp} and the communication time t_{ij}^{comm}, i.e.,

$$t_{ij} = t_{ij}^{comp} + t_{ij}^{comm} \tag{1}$$

and the completion time c_{ij} for job t_i on resource s_j equals the ready time r_j plus the offloading time t_{ij}, i.e.,

$$c_{ij} = r_j + t_{ij} = r_j + t_{ij}^{comp} + t_{ij}^{comm} \tag{2}$$

In this way, both the computation cost and the communication cost for off-loading a job can be modeled in terms of time consumption. Thus, we unified the metric for performance consideration with c_{ij}.

On different resources, the same job t_i exhibits various time consumption. We further define P_{min} and P_{max} as the minimum and the maximum time consumption of running job t_i on all the resources satisfying the minimum trust requirement $TRUST\_MIN$ of t_i. Then we define a performance penalty function:

$$F(p_r) = 1 - (p_r - P_{min})/(P_{max} - P_{min}) \tag{3}$$

Here, p_r represents the performance cost for assigning the job on a certain resource r.

With both the trust model and the performance model, the total cost for off-loading a job t_i on resource s_j can be modeled as:

$$C(r) = w_1 * F(p_r) + w_2 * P(t_r) \tag{4}$$

Here, w_1 and w_2 are two weight factors satisfying $0 \leq w_i \leq 1, i = 1, 2$ and $w_1 + w_2 = 1$. Although it seems not reasonable to weight and normalize a trust factor with a performance factor, the rational for assigning such weight factors is that the actual values for trust and for performance are normalized in $P(t_r)$ and $F(p_r)$ respectively already. In other words, the values that are to be normalized in equation 4 are normalized values for trust and performance respectively. The definition of $C(r)$ only handles the relative weight between trust and performance. In other words, it only represents how much a user weights trust over performance. In extreme cases, if a user values trust significantly higher than performance, the job is more likely to be assigned locally instead of a mobile resource. Thus, the performance may be extremely low. However, if a user pays more attention to performance, i.e. $w_1 >> w_2$, the job can be finished early by off-loading it to a powerful resource though it may have a very low trust value. A lower w_2 value gives the assigner much more choices for off-loading.

6 Analysis and Evaluation

In this section, we analyze EA-GSS from two perspectives: performance and security (including privacy). For performance issue, we calculate the computation and communication overhead. We compare the results with the previous schemes. For security issue, we evaluate the security strength of EA-GSS based on the attack model in Section 3.3.

6.1 Performance Analysis for EA-GSS

For performance, we compare the computation and communication overhead of EA-GSS with CP-ABE [5], CN scheme [30], NYO scheme (the 2nd construction in [7]), YRL scheme [6] and GIE scheme [8]. We focus on the cost of decryption process because encryption is performed once by the *Group Founder* while decryption is performed by every node receiving the group session key message. We compare the number of time-consuming operations needed in each scheme.

Our experiment is conducted on a Dell D630 machine (Intel T8100 processor 2.10GHz, 1GB memory) with Ubuntu 10.04. We choose a prime number with a scale of 154 to set up a Type A pairing using PBC Library [31]. According to our experiment, pairing is the most time-consuming operation (In Table 2, the results are the average values of 50 runs.). Thus, we calculate the number of pairing operations in decryption.

Table 2
Time-consumption for operations(in milliseconds)

Op.	Pairing	Power	Multiplication	Inversion
Time	4.574	0.088	0.016	0.038

Suppose the number of attributes a decrypter has is N_{attr}. In YRL, it represents the number of lower level attributes. The total number of attributes defined in the network is N_{all}. In CP-ABE and CN, the decrypter

Table 3
Comparison of computation cost in decryption

Scheme	Anonymity Support	Number of Pairings
CP-ABE	No	$2N_{path}+1$ or 0
CN	No	$N_{path}+1$ or 0
NYO	Yes	$2N_{all}+1$
YRL	Yes	$2N_{attr}+2$
GIE	Yes	$3N_{path}$ or $3N_{part}$
EA-GSS	Yes	$2N_{path}+1$ or $2N_{part}$

can choose the attributes used in decryption since the policy is not encrypted in these schemes. Therefore, if the decrypter satisfies the policy, the computation cost is related to the number of attributes along the decryption path of the policy tree, which is denoted as N_{path}, $N_{path} \leqslant N_{attr}$. If the decrypter doesn't satisfy the policy, the costs for both schemes are 0. This is the case when the decrypter realizes his attributes cannot satisfy the policy and thus discards the ciphertext. In GIE and EA-GSS, a decrypter cannot continue with the decryption process if he doesn't have the next attribute to decrypt. When this happens, we denote that the decrypter has already decrypted N_{part} attributes, $N_{part} \leqslant N_{path}$. Due to the limitation that some schemes only support AND-gates, our comparison is conducted for the cases when a policy is constructed with attributes and AND-gates only. Detailed results are shown in Table 3. Among the four schemes supporting anonymity, GIE and EA-GSS outperforms NYO and YRL. Also, the computation cost of EA-GSS is around 2 thirds of GIE's cost.

To evaluate communication cost, we compare the size of ciphertexts for every scheme. We denote the number of attributes used in ciphertexts as N_{ciph}. For each attribute in the policy, the corresponding ciphertext consists of 2 elements from $\mathbb{G}_0$ and 1 element from $\mathbb{Z}_p$ in EA-GSS. The total size of a ciphertext is $1\mathbb{G}_1 + (2N_{ciph} + 4)\mathbb{G}_0 + N_{ciph}\mathbb{Z}_p$. This means the ciphertext consists of 1 element from $\mathbb{G}_1$, $2N_{ciph}+4$ elements from $\mathbb{G}_0$ and N_{ciph} elements from $\mathbb{Z}_p$. Detailed results are shown in Table 4. Here, the size of attribute policy in CP-ABE and CN is not considered. Among the four schemes supporting anonymity, the ciphertext size in NYO and YRL is much larger than in GIE and EA-GSS. This is because these two schemes encrypt the ciphertext for all the attributes in the network. GIE and EA-GSS are of the same order of magnitude, but EA-GSS performs better.

6.2 Security Proof

In this section, we provide the security proof of the EA-GSS scheme following the structure in [1]. Before we start, the security game described in **Security Model** will be modified. We call the game in **Security Model** as **Game1** and the modified game as **Game2**. For each adversary **adv1** in **Game1**, we define a corresponding adversary **adv2** in **Game2**.

Table 4
Comparison of ciphertext size

Scheme	Ciphertext Size
CP-ABE	$1\mathbb{G}_1 + (2N_{ciph}+1)\mathbb{G}_0$
CN	$1\mathbb{G}_1 + (N_{all}+1)\mathbb{G}_0$
NYO	$\geqslant 1\mathbb{G}_1 + (2N_{all}+1)\mathbb{G}_0$
YRL	$1\mathbb{G}_1 + (3N_{all}+2)\mathbb{G}_0$
GIE	$N_{ciph}\mathbb{G}_1 + 3N_{ciph}\mathbb{G}_0$
EA-GSS	$1\mathbb{G}_1 + (2N_{ciph}+4)\mathbb{G}_0 + N_{ciph}\mathbb{Z}_p$

6.2.1 Modified Game

Game2 consists of five steps similar to **Game1**. The steps **Setup**, **Phase1**, and **Phase 2** are the same as in **Game1**. The **Challenge** step is different in that the challenger does not choose one message to construct the ciphertext C. Instead, it outputs C_j as:

$$C_j = \begin{cases} e(g,g)^{\alpha s_j} & \text{if } b = 1 \\ e(g,g)^{\theta_j} & \text{if } b = 0 \end{cases}$$

Here, all the θ_js are randomly chosen from Z_p following independent uniform distribution.

Suppose an adversary **adv1** in **Game1** has the advantage of ϵ, the correponding adversary **adv2** in **Game2** can be constructed as: Forward all the messages between **adv1** and the challenger during **Setup**, **Phase1**, and **Phase 2**. In the **Challenge** step, **adv2** gets two messages M_0 and M_1 from **adv1** and the challenge C from the challenger. **adv2** flips a coin δ and sends $C' = M_\delta C$ to **adv1** as the challenge for **adv1** in **Game1**. **adv2** generates its guess based on the output δ' from **adv1**. If $\delta' = \delta$, then the guess is 1; otherwise, it is 0. The advantage of **adv2** then can be calculated as $\frac{\epsilon}{2}$.

In the following, we show that no polynomial adversary **adv2** can distinguish between $e(g,g)^{\alpha s_j}$ and $e(g,g)^{\theta_j}$. Therefore, no adversary **adv1** in the original game **Game1** can have non-negligible advantage in the security model.

6.2.2 Security Guarantee in Modified Game

In this section, we use the generic group model [2]. We use a simulator to model the modified game between the challenger and the adversary. The simulator chooses a random generator $g \in G_0$ and encode any member in G_0 and G_1 to a random string following two mappings: $f_0, f_1 : \mathbb{Z}_p \to \{0,1\}^{\lceil \log p \rceil}$. One additional mapping f_2 is used to convert elements in $\mathbb{Z}_p$ to string representations: $f_2 : \mathbb{Z}_p \to \{0,1\}^{\lceil \log p \rceil}$. The three mappings are invertible so that the simulator and the adversary can map between the strings and the elements of corresponding algebraic structures in both directions. Three oracles are provided to the adversary by the simulator to simulate the group operations in G_0, G_1 and the pairing. Only the string representations can be applied to the oracles. The results are returned from the simulator in the form of string representations as well. The oracles will strictly accept inputs from the same group, i.e. strict enforcement on

the input from the same group for the respective group operations. The simulator plays the role as the challenger in the modified game.

Setup. The simulator chooses G_0, G_1, e, g and random values α, β. It also defines the mappings f_0 and f_1 and the three oracles mentioned above. The simulator also chooses the public attribute parameters $I_{Pub} \in \mathbb{Z}_p$ and $T_{Pub} = f_0(\lambda) \in G_0$, where λ is a random string. The public parameters are $g := f_0(1)$, $g^\beta := f_0(\beta)$, and $e(g,g)^\alpha := f_1(\alpha)$.

Phase 1. When the adversary runs **NodeJoin** for a new user with UID, the simulator generates a random number $r_{UID} \in \mathbb{Z}_p$. It returns to the adversary with $D_{UID} = f_0((\alpha + r_{UID})/\beta)$, $X_{Pub,UID} = f_0(r_{UID})f_0(\lambda r_{Pub,UID}) = f_0(r_{UID} + \lambda r_{Pub,UID})$, $Y_{Pub} = f_0(r_{Pub})$, and $Z_{Pub,UID} = f_1(r_{UID} I_{Pub})$, here $r_{Pub,UID} \in \mathbb{Z}_p$ is a random number chosen by the simulator. When the adversary requests for a new attribute A_i that has not been used before, the simulator randomly chooses $I_i, k_i \in \mathbb{Z}_p$ and $T_i = f_0(t_i) \in G_0$ to simulate the process for setting up an attribute authority. For each attribute key request made from the adversary, the simulator computes $X_{i,UID} = g^{r_{UID}} T_i^{r_i} = f_0(r_{UID} + t_i r_i)$, $Y_i = g^{r_i} = f_0(r_i)$, and $Z_{i,UID} = e(g,g)^{r_{UID} I_i} = f_1(r_{UID} I_i)$, where r_i is a random number chosen from $\mathbb{Z}_p$. The simulator passes all these values to the adversary as the keys associated with A_i.

Challenge. When the adversary asks for a challenge, the simulator flips a coin b and chooses a random value $s \in \mathbb{Z}_p$. If $b = 1$, the simulator calculates $C = f_1(\alpha s)$, if $b = 0$, it picks a random value $s' \in \mathbb{Z}_p$ and calculates $C = f_1(s')$. In addition, it claculates $C' = g^{\beta s}$ and $C'' = Enc_{S_k}(ROOT)$. It also computes other components of the ciphertext following **Encrypt**: $C_{1,n} = f_0((I_{n-1} - I_n)h_n)$, $C_{2,n} = f_0(t_n(I_{n-1} - I_n)h_n)$, and $C_{3,n} = f_2((k_n t_n)^{-1})$, where $h_n \in \mathbb{Z}_p$ is a random number chosen by the simulator.

Phase 2. The simulator interacts with the adversary similar as in **Phase 1** with the exception that the adversary could not acquire attribute keys that can enable it to satisfy the access policy $\mathbb{A}$.

From the above game, we can see that the adversary only acquires the string representation of random values in $\mathbb{Z}_p$ and combinations of these values. We can model all the queries as rational functions and further assume that different terms always result in different string representations. As shown in [1], the probability that two terms share the same string representation is $O(q^2/p)$, where q is the number of queries made by the adversary. We assume in the rest of the proof that no such collision happens.

Now we argue that the views of the adversaries are identically distributed between the two cases when $C = f_1(\alpha s)(b = 1)$ and when $C = f_1(s')(b = 0)$. As a matter of fact, what the adversary can view from the modified game with the simulator are independent elements that are uniformly chosen and the only operation that the adversary can do on these elements is to test if two of them are equal or not. Thus, that the views of the adversary differ can only happen when there are two different terms ν_1 and ν_2 that are equal only when $b = 1$. Since αs and s' only occur in the group G_1, the results from f_1 cannot be paired. The queries by the adversary can only be in the form of an additive term. Then we have: $\nu_1 - \nu_2 = \gamma\alpha s - \gamma' s'$. By transformation, we can have $\nu_1 - \nu_2 + \gamma' s' = \gamma\alpha s$. This implies that by deliberately constructing a query $\nu_1 - \nu_2 + \gamma' s'$, the adversary may be able to get the value of $e(g,g)^{\gamma\alpha s}$. Now we prove that such a query cannot be constructed by the adversary based on the information it gets from the simulator in the game.

In fact, the information that an adversary can acquire from this game can be enumerated. We list all the values in Table 5. (This table does not include any terms related to L_{UID} as it has nothing to do with αs.) To construct the desired value, the adversary can bilinear map two elements in G_0 into one element in G_1. He can also use elements in Z_p to change the exponentials. From this table, we can easily see that to obtain a value containing αs, the adversary can pair βs and $(\alpha + r_{UID})/\beta$ to get $\alpha s + r_{UID} s$ in G_1. In fact, this is the only way to get a term containing αs. Then, by conducting the query on behalf of the users that the adversary has established in **Phase 1**, the adversary can get a polynomial $\gamma\alpha s + \sum_{UID \in U_{query}} \gamma r_{UID} s$, where U_{query} is the set of UIDs that are used by the adversary. To eliminate the second part in this polynomial, the adversary can use items in the table containing $I_{n-1} - I_n$ and r_{UID} to construct desirable polynomial. But this is not possible for the adversary under the game assumption because:

- Firstly, the adversary cannot reconstruct s from either $t_n(I_{n-1} - I_n)h_n$ or $(I_{n-1} - I_n)h_n$ since the h_ns are chosen as random values for each attribute that it is impossible to get $s = \sum_{n \in P_a}(I_{n-1} - I_n) + I_{Pub}$ from them without peeling off the h_n's, here P_a represents the set of attributes satisfying the policy;
- Secondly, the adversary cannot reconstruct s from I_{Pub} and I_i in Z_p. This is because no single user is assumed to satisfy the attribute policy that the adversary cannot reconstruct a valid attribute combination satisfying the policy. Thus, he cannot find the constitution of P_a for the equation $s = \sum(I_{n-1} - I_n) + I_{Pub}$.

Therefore, the security of the proposed scheme is proved. ■

6.3 Performance Analysis for the Trust-incorporated Off-loading Algorithm

The evaluation environment for the off-loading algorithm is set to be an area of 1500×1500 square-meters. A certain number of mobile nodes are randomly distributed within this area. The underlying communication protocol is 802.11b at a speed of 11Mbps. End-to-end communication bandwidth is calculated as: $bandwidth =$

Table 5
Query information accessible to the adversary

from G_0		
λ	β	$(\alpha + r_{UID})/\beta$
$r_{UID} + \lambda r_{Pub,UID}$	r_{Pub}	t_i
$r_{UID} + t_i r_i$	r_i	βs
$t_n(I_{n-1} - I_n)h_n$	$(I_{n-1} - I_n)h_n$	
from G_1		
α	$r_{UID} I_{Pub}$	$r_{UID} I_i$
from Z_p		
I_{Pub}	I_i	k_i
$(k_n t_n)^{-1}$		

$one-hop bandwidth/hops$. We set the radio coverage distance for each mobile device to be 150 meters, which conforms the wifi communication range in outdoor scenario. The movements of the mobile nodes follow Random WayPoint mobility model [32]. We choose a mobile node as the client node, which runs the off-loading algorithm as the assigner. The computation power of the client node is defined as 1. The computation power of the other mobile nodes follow a uniform distribution among the values in $\{1, 2, 4\}$.

For each round of simulation, we allocate 1000 jobs for off-loading decisions. The workload of each job is characterized as the amount of time needed for the client node to execute. The jobs follow a uniform distribution among $\{20, 40, ..., 180, 200\}$ in seconds. The Computation to Communication Ratio (CCR) distribution for the jobs is constructed by selecting values from $\{0.1, 0.3, 0.5, 1, 2\}$ with the probabilities $\{0.4, 0.3, 0.15, 0.1, 0.05\}$ respectively. In other words, most of the jobs are assumed to require more time for computation than communication. This probability distribution is selected to approximate the general pareto distribution proposed in [33].

To evaluate the performance for the proposed trust-incorporated off-loading algorithm, we calculate the performance improvement over local execution under different weigh between performance concerns w_1 and trust factors w_2. The results are shown in Figure 1.

In this figure, each curve corresponds a different number of mobile devices in the simulation environment. It can be seen that the more devices in the network, the better performance can be achieved. This is reasonable as with more mobile devices, there are more candidates available as off-loading assignees. Thus, it is more likely to find a better assignee than when a very limited number of mobile devices are available.

For a single curve, i.e. when the number of mobile devices is fixed, it shows that the larger the w_1 value is, the better the overall performance can be achieved. This is mainly because with more weights put on the performance concerns as in equation 4, the decision algorithm pays more attention to find a more powerful device for task-offloading with less threshold on the trust

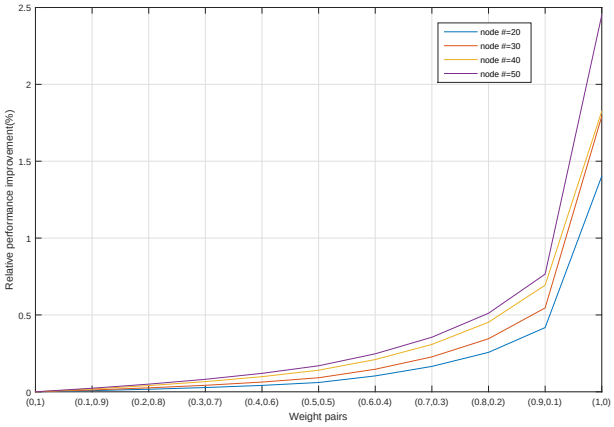

Figure 1. Off-loading performance evaluation.

of the device. In other words, the larger the w_1 value is, the more available powerful mobile devices exist for off-loading.

Based on the above observation, it is clear that the way how a trust-incorporated off-loading algorithm influences the overall performance is highly dependent on the number of mobile devices available as candidate assignees under the current trust requirement. However, such number is not tightly coupled with either w_1 or the number of nodes in network. For the latter, it can be shown from the figure that the performance of the 30-node scenario is close to that of 40 nodes in our experiment when w_1 gets close to 1.0.

7 Conclusion

In this paper, we propose the EA-GSS scheme for secure and trustworthy task off-loading in mobile cloud. In EA-GSS, trust and performance are both considered as important factors for off-loading decisions. To set up trust groups, there is no need for any prior knowledge of other group members beforehand. The group formation policies are preserved in a gradual exposure way which greatly reduces the computation and communication overhead compared to existing hidden-policy solutions. Moreover, EA-GSS is designed to support flexible group management, which is quite suitable for off-loading in mobile cloud environment. Forward and backward secrecies are achieved using a key update mechanism. Experiments and analysis confirm its effectiveness in improving the performance for task execution while keeping a reliable trust guarantee.

Acknowledgment

This research is sponsored by ONR YIP Award N00014-10-1-0714 and ARO Research Grant W911NF-11-1-0191.

References

[1] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui:

Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 2010, pp. 49–62.

[2] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "Comet: Code offload by migrating execution transparently," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, 2012, pp. 93–106.

[3] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *INFOCOM, 2012 Proceedings IEEE*, 2012, pp. 945–953.

[4] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Comput. Surv.*, vol. 35, pp. 309–329, 2003.

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, 2007, pp. 321–334.

[6] S. Yu, K. Ren, and W. Lou, "Attribute-based on-demand multicast group setup with membership anonymity," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, 2008, pp. 18:1–18:6.

[7] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Proceedings of the 6th international conference on Applied cryptography and network security*, 2008, pp. 111–129.

[8] D. Huang, Z. Zhou, and Z. Yan, "Gradual identity exposure using attribute-based encryption," in *International Conference on Social Computing*, 2010, pp. 881 –888.

[9] M. Kristensen, "Scavenger: Transparent development of efficient cyber foraging applications," in *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, 2010, pp. 217–226.

[10] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *In ACM MobiHoc*, 2012.

[11] C. Shi, M. H. Ammar, E. W. Zegura, and M. Naik, "Computing in cirrus clouds: The challenge of intermittent connectivity," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012, pp. 23–28.

[12] M. Shiraz and A. Gani, "A lightweight active service migration framework for computational offloading in mobile cloud computing," *The Journal of Supercomputing*, pp. 978–995, 2014.

[13] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, pp. 14–23, 2009.

[14] S. Ou, K. Yang, and J. Zhang, "An effective offloading middleware for pervasive services on mobile devices," *Pervasive Mob. Comput.*, pp. 362–385, 2007.

[15] M. Shiraz, E. Ahmed, A. Gani, and Q. Han, "Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing," *The Journal of Supercomputing*, pp. 84–103, 2014.

[16] T. Verbelen, T. Stevens, F. De Turck, and B. Dhoedt, "Graph partitioning algorithms for optimizing software deployment in mobile cloud computing," *Future Gener. Comput. Syst.*, pp. 451–459, 2013.

[17] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in *Proceedings of the Eighth Heterogeneous Computing Workshop*, 1999, pp. 30–.

[18] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, pp. 810–837, 2001.

[19] Y. Qin, D. Huang, and B. Li, "Stars: A statistical traffic pattern discovery system for manets," *Dependable and Secure Computing, IEEE Transactions on*, pp. 181–192, 2014.

[20] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1998, pp. 85–97.

[21] C. Sarr, C. Chaudet, G. Chelius, and I. Lassous, "Bandwidth estimation for ieee 802.11-based ad hoc networks," *Mobile Computing, IEEE Transactions on*, pp. 1228–1241, 2008.

[22] L. Chen and W. Heinzelman, "Qos-aware routing based on bandwidth estimation for mobile ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, pp. 561–572, 2005.

[23] J. Li, C. Blake, D. S. De Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001, pp. 61–69.

[24] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 2010, pp. 693–702.

[25] D. Huang, Z. Zhou, and Z. Yan, "Gradual identity exposure using attribute-based encryption," in *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, 2010, pp. 881–888.

[26] B. Li, A. Verleker, D. Huang, Z. Wang, and Y. Zhu, "Attribute-based access control for icn naming scheme," in *Communications and Network Security*

(CNS), 2014 IEEE Conference on, 2014, pp. 391–399.

[27] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," in *The Annals of Mathematical Statistics 38 (1967)*.

[28] N. Li, J. Mitchell, and W. Winsborough, "Design of a role-based trust-management framework," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, 2002, pp. 114–130.

[29] S. Ruohomaa and L. Kutvonen, "Trust management survey," in *Proceedings of the Third International Conference on Trust Management*, 2005, pp. 77–92.

[30] L. Cheung and C. Newport, "Provably secure ciphertext policy abe," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 456–465.

[31] B. Lynn, "Pbc library," in *http://crypto.stanford.edu/pbc/*, Accessed March 2013.

[32] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing (WCMC)*, pp. 483–502, 2002.

[33] D. G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation*. Cambridge University Press, 2015.