Hybrid Attribute-Based Encryption and Re-Encryption for Scalable Mobile Applications in Clouds

Piotr K. Tysowski and M. Anwarul Hasan
Dept. of Electrical & Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada
pktysowski@uwaterloo.ca and ahasan@uwaterloo.ca

Abstract

Outsourcing data to the cloud is beneficial for reasons of economy, scalability, and accessibility, but significant technical challenges remain. Sensitive data stored in the cloud must be protected from being read in the clear by a cloud provider that is honest-but-curious. Additionally, cloud-based data is increasingly being accessed by resource-constrained mobile devices for which the processing and communication cost must be minimized. Novel modifications to attribute-based encryption are proposed to allow authorized users access to cloud data based on the satisfaction of required attributes such that the higher computational load from cryptographic operations is assigned to the cloud provider and the total communication cost is lowered for the mobile user. Furthermore, data re-encryption may be optionally performed by the cloud provider to reduce the expense of user revocation in a mobile user environment while preserving the privacy of user data stored in the cloud. The proposed protocol has been realized on commercially popular mobile and cloud platforms to demonstrate real-world benchmarks which show the efficacy of the scheme. A simulation calibrated with the benchmark results shows the scalability potential of the scheme in the context of a typical workload in a mobile cloud computing system.

1 Introduction

Cloud computing offers the advantages of highly scalable and reliable storage on third-party servers. Its economical pay-per-use model typically results in a small fraction of the cost of deploying the same computing resources in-house. Data outsourcing to a cloud is appropriate for any class of applications that requires data to be kept in storage and disseminated to many users. Clients that engage a cloud provider typically only pay for the amount of storage, related computation, and amount of network communication actually consumed; they do not incur the capital and maintenance costs of an in-house solution. In addition, the cloud provider offers the advantages of automatic backup and replication to ensure the safety, longevity, and high accessibility of the user data. A major concern that is typically not sufficiently addressed in practice, however, is that data, by default, is stored in the clear; it may be accessed and read by a cloud administrator without knowledge of the client. A cloud administrator may not be trusted despite the presence of contractual security obligations, if data security is not further enforced through technical means. An additional risk is that sensitive data carries the persistent risk of being intercepted by an unauthorized party and possibly manipulated, despite safeguards promised by the provider. Therefore, it is useful to apply software techniques, such as encryption keys, to ensure that the confidentiality

of cloud data is preserved at all times. It is especially crucial to safeguard sensitive user data such as e-mails, personal customer information, financial records, and medical records.

It must also be recognized that the recent trend in contemporary cloud computing applications is for cloud data to be accessed primarily by resource-constrained mobile devices, a practice known as *mobile cloud computing*. This device category includes users of smartphones and tablets; in some applications, it is appropriate to consider smart wireless sensors in the same mix. Hence, any protocol providing additional security must not add burdensome costs to a mobile user; specifically, the number of transmissions must be minimized to conserve the battery and over-the-air data usage fees, and the amount of computation must also be minimized to avoid adding significant delays to the user experience while further decreasing battery life. Another important requirement is for data to be addressable with fine-grained access controls on the record-level or finer, to provide flexibility. A single user log-in is largely insufficient in today's complex data retrieval.

The main contributions of the proposed work are as follows:

1. A protocol for outsourcing data storage to a cloud provider in secure fashion is provided. The cloud provider is unable to read stored data; authorized users may do so based on qualification through possession of the right attributes without arbitration by the data owner. The protocol is designed to be efficient for resource-constrained mobile users by delegating computation and requests to a cloud provider or trusted authority, where appropriate, without compromising security.

2. An improvement is made over a traditional attribute-based encryption scheme, such that responsibility over key generation is divided between a mobile data owner and a trusted authority; the owner is relieved of the highest computational burden.

3. Additional security is provided through a group keying mechanism; the data owner controls access based on the distribution of an additional secret key, beyond possession of the required attributes. This additional security measure is an optional variant applicable to highly sensitive data subject to frequent access.

4. Re-encryption, as a process of transforming the stored ciphertext, permits efficient revocation of users; it does not require removal of attributes and subsequent key regeneration, and may be administered by a trusted authority without involvement of the data owner.

In Section 2, related work on key management to secure cloud data storage is presented, with a focus on attribute-based schemes in the context of applications accessed by mobile devices. In Section 3, a system model encompassing a mobile cloud computing system is presented, as well as a model of trust that is assumed for its security. In Section 4, the proposed algorithm for attribute-based encryption and re-encryption suitable for mobile users of the cloud is presented. In Section 5, optional features of the algorithm such as delegation are presented, for completeness. In Section 6, the algorithm is assessed for its usefulness in a mobile cloud computing system. In Section 7, the results of an implementation of the proposed scheme on actual mobile devices and an operational cloud system are presented and discussed; a simulation is then used to demonstrate its scalability potential. Finally, Section 8 provides concluding remarks.

2 Related work

Numerous solutions may be envisaged to exchange encrypted data with a cloud provider in a secure manner, such that the cloud provider is not directly entrusted with key material, but naïve

schemes often prove difficult to scale. For instance, the main drawback of a scheme based on the use of a public key management system using RSA is that it requires that the data owner provide an encrypted version of data for each recipient that may access the same data, which becomes impractical once the system scales significantly. If user data is encrypted with a single key, then that key must be shared with all authorized users, which carries a high traffic cost especially if this obligation rests on a mobile data owner.

Users may join and leave the authorized user set frequently, leading to constant key re-generation and re-distribution through additional communication sessions to handle user revocation; in a highly scaleable system composed of thousands of users, such events may occur at relatively high frequency. Wireless communication, however, is expensive and results in rapid battery drain, particularly when transmitting [1].

The technique of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [2] offers numerous advantages in the envisioned target environment. It allows a user to obtain access to encrypted data in the cloud based on the possession of certain attributes that satisfy an access structure defined in the cloud, rather than the possession of a particular individual or group key that must be disseminated to all interested parties in advance. The requisite attributes may be determined by a data owner in advance; this owner is responsible for generating the user data to be shared, encrypting it, and uploading it to the cloud. Unauthorized access to stored data is not in itself an issue due to the protection afforded by CP-ABE. Furthermore, the data owner is not required in every data transaction involving other users, which is advantageous in the case where "always-on" connectivity cannot be guaranteed. It is impossible for any two users to collude by combining their individual attributes to gain access that would otherwise not have been individually granted. Normally, a scheme based on CP-ABE relies upon the data owner granting access permission through an access tree, which requires some level of availability. Some works have modified CP-ABE so that key material is distributed among multiple parties; for instance, a data owner and a trusted authorizer may function in concert to grant access permission to other users, building on the OAuth standard [3]; the solution, however, is not tailored for a mobile environment due to its computational demands, constant availability of the data owner that is required, and use of time-based expiration of access that leads to frequent key retrieval.

Revocation of an authorized user is particularly difficult to accomplish efficiently in CP-ABE and is usually addressed by extending attributes with expiration dates or by an authority distributing keys with expiration dates [2]. In some cases, a tree of revocable attributes may need to be maintained and a trusted party assigned to validate the revocation statuses of users; the access control may be system-wide or more fine-grained. A revocation mechanism using linear secret sharing and binary tree techniques, where each user is associated with an identifier on a revocation tree, is one example [4]. The difficulty with this general approach in a mobile context is that it results in mobile users having to incur the communication cost of continually requesting new keys, while wireless communication always remains expensive. Also, the data owner is typically a mobile user as well, and thus the owner cannot effectively manage access control on demand for other users due to its transient connectivity. Revocation for data outsourcing purposes has been proposed that relies on stateless key distribution and access control on the attribute level, but requires a trusted authority and encumbers the data owner with a pairing operation [5], a cryptographic function that is very computationally expensive.

To handle revocation in a highly scalable system, a scheme has been proposed that uses the cloud provider for distributing portions of key material and for automatic and blind data re-encryption [6], such that the plaintext cannot be recovered in the process. As with many other schemes, it relies upon the presence of a trusted manager which is an additional network component that must be scaled and maintained itself. While effectively handling changes in user memberships,

its limitation is that it requires granting access solely based on user identity, as opposed to the attributes held by users. Various other proxy re-encryption schemes have been applied to secure distributed storage. For instance, an access control server may securely store content keys within a lockbox that is re-encrypted on-demand to allow an authorized user to access the content keys, although it requires the data owner to be present and aid in the re-encryption task [7].

As a next step in the evolution of such techniques, proxy re-encryption has been combined with CP-ABE [8] such that re-encryption keys are computed by the cloud provider based on a secret that is pre-shared between the data owner and the provider, as well as the provider's internal clock. The re-encryption keys must be computed for all attributes in the access structure, which could be very numerous. Another idea is to securely embed the data key within the header of the record stored in the cloud [9]; a privileged manager group is responsible for generation of re-encryption keys, but it must also distribute the secret header key to the recipient to complete the process.

3 Model of mobile cloud computing

3.1 System model

A *CSP* (Cloud Service Provider), simply referred to as a "cloud," provides permanent data storage in a centralized data centre or a small host of geographically dispersed but interconnected centres. The user data stored in the cloud may be directly accessed by users over the public Internet infrastructure by referencing a particular data partition.

On the client end, many users will operate portable devices such as smartphones or tablets that are significantly more limited than desktop computers in terms of on-board memory, processors, useful operating life, and available network bandwidth. Due to the nature of wireless networks, mobile users will likely suffer from transient connectivity such that their constant availability in the system cannot be guaranteed.

The system model of the proposed work is shown in Figure 1. Inside the public cloud, a *controller* administrates access through external client interfaces. Requests, including data uploads and downloads, are made over the reliable but insecure medium of the Internet; a wireless packet data infrastructure serves to bridge mobile users. The manager is a trusted self-supporting network component that may be situated behind an organization's firewall and form part of a private cloud belonging to the client. It may maintain a database of private key information relating to a set of authorized mobile users. Inside the cloud, the controller maintains a complementary public key information database, and stores and reads user data on behalf of clients to and from the permanent and replicated data store. The user data may periodically undergo cryptographic transformation, such as re-encryption from one version of ciphertext to another; such activities are dispatched on-the-fly or alternatively at off-peak times by eligible worker processes initiated by the controller.

A mobile user may act as a *data owner* and decide what access privileges are appropriate for the data that it uploads to the cloud and retains control over; a specific subset of the user population may be identified as having sufficient permission based on unique identities, or users may be assigned various distinguishing attributes that inherently grant permission regardless of the specific identity that assumes them. A highly scalable system is envisioned where users may number potentially in the thousands or millions. Continuous arbitration by a single data owner during all transactions is impractical, as a mobile user is subject to a limited battery and transient connectivity.

Possible applications of the secure data outsourcing approach suggested here include highly-collaborative enterprise services, secure data storage and retrieval, and social networking sites.

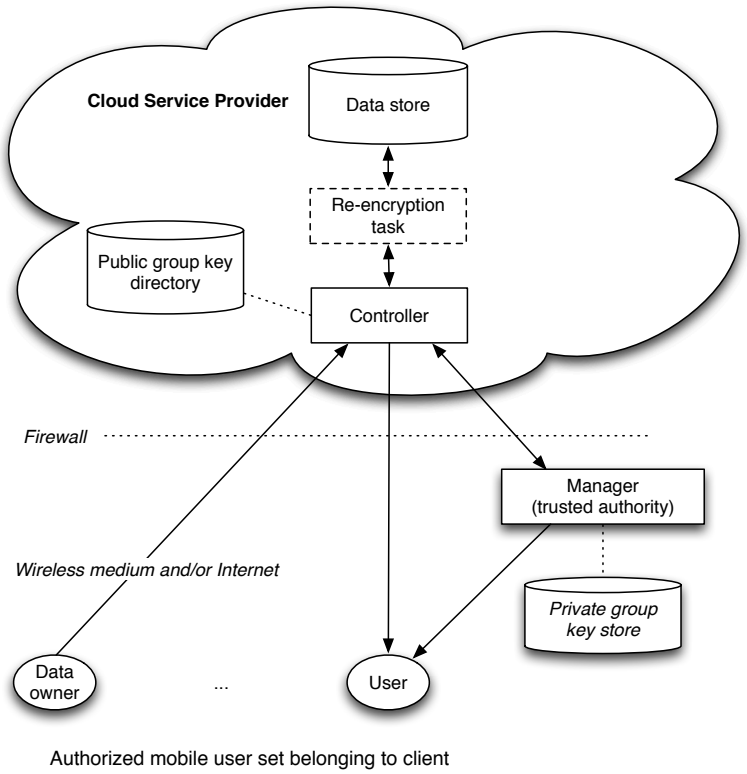

Figure 1: System model.

3.2 Trust model

The cloud provider is assumed to be honest-but-curious in practice; it will generally obey a communications protocol and deployed application logic, and will not deny service to any authorized party. At the same time, a cloud administrator may read the contents of user data stored in the cloud for nefarious reasons or simply out of curiosity. A party with administrator privileges may even copy or modify data without the client's knowledge. Thus, data stored in the cloud should remain encrypted at all times, and any required transformation of it should not reveal the plaintext in the process. The communications channel between a user and the cloud is open and subject to eavesdropping; thus, sensitive user data may not be exchanged in the clear.

The manager is a trusted authority within the system and is administered under the domain of the users in question; it is completely independent of the CSP. It is sufficiently trusted to authorize access to the cloud and to contain key material as necessary; however, to minimize the risk of it being compromised, a user will only share as much of its own key material with the manager as is necessary in the security scheme utilized. Furthermore, the manager will not be as economical as a cloud provider due to its more limited computational resources.

4 Proposed algorithm

The proposed algorithm for key generation, distribution, and usage is now described. It consists of key management techniques that ensure highly secure data outsourcing to the cloud in a highly scaleable manner for mobile cloud computing applications. Table 4 summarizes the symbolic notation used throughout the description.

Symbol	Description
CSP	Cloud service provider.
M	Trusted manager.
T	Access tree structure.
R	Root node of T.
A	Set of attributes that must be satisfied against T.
U_o	Data owner.
U_r	Restricted user group.
m	Plaintext of user data.
CT	Ciphertext of user data.
v	Version of ciphertext.
PPK	Public partition key.
PSK	Secret partition key.
OPK	Owner public key.
OSK	Owner secret key.
DSK	Data secret key.
GPK	Public group key of U_r.
GSK	Private group key of U_r.
$DDSK$	Delegated data secret key.
$RK_{0\to x}$	Re-encryption key from version 0 to x.

Table 1: Legend for symbolic notation.

Improvements are proposed to the basic functions of the original CP-ABE scheme [2] as follows:

- A single authority does not generate all key material; the mobile data owner and cloud entity co-operate to jointly compute keys. The cloud provider has insufficient information to decode the user data that it permanently stores; yet, it assists in the distribution of a portion of the whole key material to all authorized users to minimize the communication cost for the data owner.
- The cloud has highly scalable computational ability, unlike a resource-constrained mobile user. A trusted manager is also better equipped than a user. Pairing operations, which are the most expensive cryptographic operations that are involved in the proposed protocol, are thus performed by the cloud or manager entities to the maximum possible extent, relieving the burden on the mobile data owner.
- Proxy-based re-encryption has been integrated with CP-ABE so that the cloud provider may perform automatic data re-encryption; this is an optional feature that allows further control over revocation than is afforded by an attribute-based scheme alone, and it also takes advantage of the cloud provider's computational scalability. This dual-encryption scheme is a hybrid approach that offers greater flexibility in access control.

The proposed technique is now described as follows:

Preliminary:

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be cyclic bilinear groups of prime order p with generator g. Also defined are random exponents $\alpha, \beta \in \mathbb{Z}_p^*$. The bilinear map e is a map such that: $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ with the following properties:

- Bilinearity: $\forall g_0, g_1 \in \mathrm{G}_0 : e(g_0^\alpha, g_1^\beta) = e(g_0, g_1)^{\alpha\beta}$.

- Non-degeneracy: $e(g_0, g_1) \neq 1$.
- Computability: $\forall g_0, g_1 \in \mathrm{G}_0$, there is an efficient algorithm to compute $e(g_0, g_1)$.

A secure one-way hash function $H : 0, 1^* \rightarrow \mathbb{G}_0$ is used as a random oracle and maps an attribute described as a binary string to a random group element.

Setup$() \rightarrow PPK, PSK, OSK$:

Suppose that Alice is a mobile user that acts as the self-elected data owner U_o of plaintext message m, which is user data that is desired to be encrypted and shared in the cloud with other authorized users.

If m exceeds the maximum allowed block length, then two solutions are possible: segmentation of the message may be performed, and the encryption applied to each individual segment; or, it is possible to first apply a symmetric cipher such as 256-bit AES to the entire message, then to encrypt the AES key itself using the proposed scheme, with the steps being reversed on decryption. Regardless, the length of the message does not impact the size or number of encryption keys required. In the case of message segmentation, the same pre-computed keys may be applied to all segments.

A manager M, acting as a trusted entity, chooses a random value α and computes g^α to form a private partition key PSK. It then performs a pairing operation to compute component $e(g, g)^\alpha$, which becomes one of the components forming the public partition key PPK. The public parameters $\mathbb{G}_0$ and g are also included in PPK. In the meantime, U_o chooses a secret data owner key OSK equal to β. It then computes the components g^β and $g^{\frac{1}{\beta}}$, the latter by first taking the inverse of OSK (i.e. $\frac{1}{\beta}$) in its possession; these components are added to the public key PPK, which is then uploaded and published in the public directory of the cloud. U_o does not divulge its secret OSK to any other party, including M. The elements of PPK, PSK, and OSK are as follows:

$$PPK = \left\{\mathbb{G}_0, g, g^\beta, g^{\frac{1}{\beta}}, e(g, g)^\alpha\right\}$$
$$PSK = \{\alpha, g^\alpha\}, \quad OSK = \{\beta\}$$

To provide an additional layer of security, an individual user or a restricted subset of users U_r may create their own shared group key GSK equal to a random value $u_0 \in \mathbb{Z}_p^*$, and a public key GPK as follows:

$$GPK = \{g^{u_0}\}, GSK = \{u_0\}$$

This group key is uploaded to the public directory as well. The secret key is not shared with the cloud; it may however be shared with the manager for the purpose of distribution to all authorized users. The initial version number of the secret key is initially referred to as 0, and will increase monotonically.

Encrypt$(PPK, GPK, m, T) \rightarrow CT$:

Any user may access the public partition key PPK, by downloading it from the public directory in the cloud, to perform an encryption; it need not necessarily be the data owner.

The encryption algorithm takes as input the key PPK and encrypts a message m under the tree access structure T with root R as described in [2]. It chooses a polynomial q_x for each node x in T, and a random value $s \in \mathbb{Z}_p^*$ that is applied to the PPK parameters. It sets $q_R(0) = s$ for the root node R, while Y denotes the set of leaf nodes in T. The function $\gamma(y)$ extracts the binary attribute string from a leaf node y in Y.

In order to protect highly sensitive data, the encryptor may wish to restrict user membership requirements beyond possession of the required attributes A. To do so, an additional key component may be incorporated consisting of $e(g,g)^{u_0 s}$, computed from g^{u_0}, which is the public key GPK of a restricted user group U_r belonging to the entire population of users. The group consists of one or more members, and is available from the public directory in the cloud. The absence of this component, where u_0 is presumed to be nil, will allow decryption based on satisfaction of the access tree only.

The intermediate ciphertext CT_{own} is constructed as follows by the data owner U_o and uploaded to the cloud:

$$\begin{aligned} CT_{own} = \Big\{ & v = 0, T, C_{0_{msg}} = m \cdot e(g,g)^{\alpha s}, C_{0_{grp}} = g^{u_0 s}, \\ & C' = g^{\beta s}, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(\gamma(y))^{q_y(0)} \Big\} \end{aligned} \tag{1}$$

Next, the CSP performs a pairing operation on the $C_{0_{grp}}$ component in CT_{own} to obtain the result $\hat{C}_{0_{grp}} = e(g,g)^{u_0 s}$. The final ciphertext CT_0, denoting the initial version v of 0, is constructed as follows and published in the permanent data store of the cloud:

$$\begin{aligned} \tilde{C}_0 &= C_{0_{msg}} \cdot \hat{C}_{0_{grp}} = m \cdot e(g,g)^{\alpha s} \cdot e(g,g)^{u_0 s} = m \cdot e(g,g)^{\alpha s + u_0 s} \\ CT_0 &= \Big\{ v = 0, T, \tilde{C}_0, C_{0_{msg}}, C' = g^{\beta s}, \forall y \in Y : C_y, C'_y \Big\} \end{aligned} \tag{2}$$

Re-Encrypt$(CT_0, RK_{0 \to x}) \to CT_x$:

Whenever a user leaves the authorized membership of U_r, the user's access rights to the ciphertext must be revoked. When this occurs, a new version of the secret group key GSK is normally distributed by the manager to the remaining authorized users in U_r, or distributed to each user on-demand through a secure off-line channel whenever data access is required.

The CSP is then requested to perform a re-encryption operation on-demand so that its stored ciphertext can no longer be decoded using the prior version of the key. The ciphertext is re-encrypted from version 0 to version x, given a re-encryption key $RK_{0 \to x}$ from a user holding group secret key GSK_x assigned to version x; or, $RK_{0 \to x}$ may be transmitted by the manager which is entrusted with the safekeeping of the key GSK_x. The re-encryption key is computed from the secret group key values u_0 and u_x corresponding to versions 0 and x of the ciphertext:

$$RK_{0 \to x} = \Big\{ g^{\frac{u_x}{u_0}} \Big\}$$

The cloud provider computes the new ciphertext CT_x corresponding to version x (that is newer than the original version 0 uploaded by the encryptor), as follows, utilizing the component $C_{0_{msg}}$ found in CT_0 in Equation 2:

$$\begin{aligned} \tilde{C}_x &= C_{0_{msg}} \cdot e(C_{0_{grp}}, RK_{0 \to x}) = m \cdot e(g,g)^{\alpha s} \cdot e(g^{u_0 s}, g^{\frac{u_x}{u_0}}) = m \cdot e(g,g)^{\alpha s + u_x s} \\ CT_x &= \Big\{ v = x, T, \tilde{C}_x, C_{x_{base}} = g^{u_x s}, C', \forall y \in Y : C_y, C'_y \Big\} \end{aligned}$$

The CSP is unable to decode the ciphertext during the re-encryption process as it has no knowledge of the old key u_0 and the new key u_x. The cloud provider retains the component $C_{x_{base}}$ in the ciphertext CT_x so that it may perform a future re-encryption from version x to y, where $y > x$.

KeyGen$(PPK, PSK, A) \rightarrow DSK$:

Irrespective of which party performed the encryption, the manager executes a data secret key generation algorithm which takes as input the private key PSK and a set of attributes A that are deemed sufficient to decrypt the ciphertext. Specifically, the manager chooses a random $r \in \mathbb{Z}_p^*$ and computes $(\alpha + r)$; it then exponentiates the component $g^{\frac{1}{\beta}}$ in the PPK by this sum to obtain the result $g^{\frac{(\alpha+r)}{\beta}} = (g^{\frac{1}{\beta}})^{\alpha+r}$. In this way, during the collaboration, the manager and data owner do not need to reveal their private keys PSK and OSK to one another. The data owner is not involved in the key generation and need not remain available.

To generate the additional required sub-parts of the data key, the manager chooses random $r_j \in \mathbb{Z}_p$ for each attribute in A. It computes the data secret key DSK that identifies with the attributes A as follows:

$$DSK = \left(D = g^{\frac{(\alpha+r)}{\beta}}, \forall j \in A : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j} \right) \tag{3}$$

The manager distributes a DSK based on a unique r value to each authorized user holding the required attributes A. The manager may also provide the DSK to the data owner for peer-to-peer distribution at its discretion to the intended recipients of the encrypted message, without requiring the participation of the data owner.

Decrypt$(CT, DSK, PPK, GSK) \rightarrow m$:

Any user that is authorized, by virtue of holding the required attributes, may download the ciphertext CT from the cloud and decrypt it, as the recipient. The decryption routine takes as input the ciphertext CT and data secret key DSK obtained earlier either from the manager M or data owner U_o. The recursive decryption algorithm DecryptNode is applied to the root node R of the tree T that is publicly available on the cloud for download.

If the node x is a leaf node, then let $i = \gamma(x)$, where the function γ denotes the attribute associated with the leaf node x in T. If $i \in A$, then the DecryptNode function is defined as follows, using components D_i and D'_i derived from the DSK, as found in Equation 3, and C_x and C'_x derived from CT_{own}, as found in Equation 1:

$$\begin{aligned} \text{DecryptNode}(CT_{own}, DSK, x) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} = \frac{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ &= \frac{e(g^r \cdot g^{\delta r_i}, g^{q_x(0)})}{e(g^{r_i}, g^{\delta q_x(0)})} = e(g,g)^{rq_x(0)} \end{aligned}$$

The recursive case, when x is a non-leaf node, is described in detail in [2]. If the access tree is satisfied by attributes A (that determined the data secret key DSK), observe that the DecryptNode function gives the following result:

$$\text{DecryptNode}(CT_{own}, DSK, R) = e(g,g)^{rq_R(0)} = e(g,g)^{rs}$$

If the ciphertext is optionally encoded with the public key of the restricted user group U_r, then the recipient may utilize the secret key $GSK = u_x$ in conjunction with the $g^{\frac{1}{\beta}}$ component in the PPK to compute the required decryption component $g^{\frac{u_x}{\beta}}$.

The message m can then be decrypted as follows:

$$m = \frac{C \cdot e(g,g)^{rs}}{e(g^{\beta s}, D) \cdot e(g^{\beta s}, g^{\frac{u_x}{\beta}})} = \frac{m \cdot e(g,g)^{\alpha s + u_x s} \cdot e(g,g)^{rs}}{e(g^{\beta s}, g^{\frac{(\alpha+r)}{\beta}}) \cdot e(g^{\beta s}, g^{\frac{u_x}{\beta}})} = \frac{m \cdot \cancel{e(g,g)^{(\alpha+r+u_x)s}}}{\cancel{e(g,g)^{(\alpha+r+u_x)s}}}$$

Note that the component $e(g^{\beta s}, g^{\frac{(\alpha+r)}{\beta}})$ in the above equation is pre-computed by the manager, so that the user must perform only one pairing operation on decryption.

A summary of the key material in possession within the system is given in Table 4. The cryptographic operations described in this section, applied to a typical encryption and decryption transaction, are summarized in Table 5.

Entity	**Key material**
Data owner (U_o)	Chooses random β. Computes: $OSK = \{\beta\}$, $OPK = \{g^{\beta}, g^{\frac{1}{\beta}}\}$ Chooses random u_o. Computes: $GSK = \{u_0\}$, $GPK = \{g^{u_0}\}$ Shares GSK with user B. Chooses random s. Computes CT_{own} based on the PPK and GPK, and uploads it to CSP.
Manager (M)	Chooses random α. Computes: $PSK = \{\alpha, g^{\alpha}\}$, $PPK = \left\{\mathbb{G}_0, g, g^{\beta}, g^{\frac{1}{\beta}}, e(g,g)^{\alpha}\right\}$ Chooses random r. Computes: $D = g^{\frac{(\alpha+r)}{\beta}}$, and DSK based on attributes A. Distributes DSK to user B.
Cloud (CSP)	Optionally performs a pairing operation and computes CT_0 from CT_{own} and GPK. Stores CT_0 in the permanent data store.
Recipient (B)	Downloads CT_0 and decrypts it using the DSK and GSK.

Table 2: Summary of key material in the proposed protocol.

5 Optional features

The original CP-ABE scheme defined delegation, where following the generation of the data secret key DSK, the manager may choose to delegate access to a user possessing a particular subset of the required attributes. For completeness, the integration of this optional operation with the proposed technique is shown:

Delegate$(DSK, S') \rightarrow DDSK$:

The delegation algorithm first takes as input a secret data key for a set of attributes A, and a subset $A' \subseteq A$. The manager chooses a new random $r' \in \mathbb{Z}_p^*$ and computes the result:

$$D' = D \cdot (g^{\frac{1}{\beta}})^{r'}$$

The manager then chooses a new random r'_k for each attribute k in the subset S' to form the next sub-part, and creates a new secret delegation data key $DDSK$ for S':

$$DDSK = \left(D' = D \cdot (g^{\frac{1}{\beta}})^{r'}, \forall k \in A' : D'_k = D_k \cdot g^{r'} \cdot H(k)^{r'_k}, D''_k = D'_k \cdot g^{r_k}\right)$$

Since the delegation key is randomized with the value r', it is equivalent in its level of security to the original key DSK. Note that the data owner is not involved in this operation.

6 Discussion

The proposed scheme offers a dual layer of security through attribute-based encryption and also public key encryption which may be optionally applied. If the secret group key GSK is compromised, the data is still safeguarded; only the users that have the required attributes will be able to

	Alice (U_o)	**Cloud** (CSP)	**Manager** (M)	**Bob** ($\in U_r$)
1	Generates private owner key OSK and sends public component OPK to M to form partition key PPK. Generates private and public group keys GSK and GPK to share with trusted users and the CSP.	Stores partition key PPK obtained from M in a public directory for dissemination to all authorized users. Also stores the public group key GPK that it obtains from Alice.	Generates private and public partition keys PSK and PPK, the latter with assistance from Alice, and uploads PPK to CSP.	Obtains GSK from Alice as a trusted user.
2	Assuming that Alice is also the encryptor, encrypts message m, with PPK and under tree T, as CT_{own}, and uploads it to CSP for storage. Also generates a component for data key DSK from OSK.	Computes CT_o from CT_{own} and GPK, and stores it as ciphertext version 0 in permanent storage, for dissemination to all authorized users.	Generates data key DSK from its private partition key PSK based on attributes A, with assistance from Alice. Distributes the DSK to all authorized users.	Obtains the DSK from Alice or M.
3				Downloads CT_o from CSP and decrypts it to yield the plaintext m.

Table 3: Summary of operations with participating user actors.

decrypt it. The interception of any key components over the network, including the re-encryption key, will not yield useful information to the attacker, as no private keys are transmitted in the clear. Furthermore, the algorithm achieves collusion resistance because the $e(g,g)^{\alpha s}$ term of ciphertext CT cannot be recovered by an attacker even if the manager's or a user's private keys are compromised.

Any user may encrypt data using the public partition key PPK stored in the cloud. However, the cloud provider is unable to decrypt any user data stored on its premises as it cannot access the secret owner and data keys OSK and DSK. Nor is useful information revealed to the CSP during the re-encryption process.

The encryptor of a message may restrict its eligible readership by not only selecting a required set of attributes, but also through the optional use of a public group key which may be shared by a group or simply possessed by a single user. The trade-off made is the required distribution of the group key and the extra pairing operation required during the encryption phase, but it is advantageously computed by the cloud provider.

Critically, the performance implications are modest for the mobile users. The data owner must only perform exponentiation operations during its key generation phases, while the manager performs the more expensive pairing operation during partition key generation in the SETUP algorithm.

If a group secret key is utilized, it may be advantageous for the manager to compute new key versions and re-encryption keys, and manage their storage and distribution. A suitable key versioning mechanism is suggested for this purpose, such as the one found in [6].

7 Implementation

The proposed protocol was implemented and profiled to gauge its performance. It was realized on popular existing commercial platforms, including the Google Android mobile and the Google App Engine cloud platforms. A simulation calibrated to the performance benchmarks was then run to examine the scalability of the proposed algorithm.

7.1 Performance measurement

An existing implementation of CP-ABE in Java [10], which relies upon the original CP-ABE scheme [2], served as the baseline implementation. From this starting point, the implementation was significantly rebuilt to reflect the proposed protocol described herein. The implementation uses the Java Pairing-Based Cryptography Library (jPBC) [11], a port of the PBC (Pairing-Based Cryptography Library) in C [12]. The use of Java 6 Standard Edition permits the protocol to be ported to a wide range of computing environments.

The implementation was run on different computing hosts to assess their relative performance. Refer to the implementation model in Figure 2. On the client end, a simulation was run on a desktop platform consisting of an Apple iMac with a quad-core 64-bit 3.4 GHz Intel Core i7 processor with 16 GB of RAM, running Mac OS X 10.8.2 (Mountain Lion). Additionally, it was run on an older Google Nexus One smartphone with a 1 GHz Qualcomm Scorpion processor with 512 MB memory running Android OS 2.3.6 (Gingerbread), and a new Samsung Galaxy Note II smartphone with a quad-core 1.6 GHz ARM Cortex-A9 processor with 2 GB of RAM, running Android OS 4.1.1 (Jelly Bean). On the server end, a lowest-class F1 front-end instance was run as a Java servlet application on the Google App Engine (GAE) cloud, configured at the equivalent of a 600 MHz processor with 128 MB of RAM. A connection was established between the desktop or mobile Android client and an instance running on the GAE cloud via HTTP requests, using JSON for data interchange and the Google Gson library for marshalling between Java objects (used by the Java client and server implementations) and the JSON representation. Note that the security model of GAE does not allow direct network connections and native code execution. Only a subset of the Java 2 Standard Edition (J2SE) SDK 1.6 classes are whitelisted on Android and GAE; fortunately, the required JCE classes are supported.

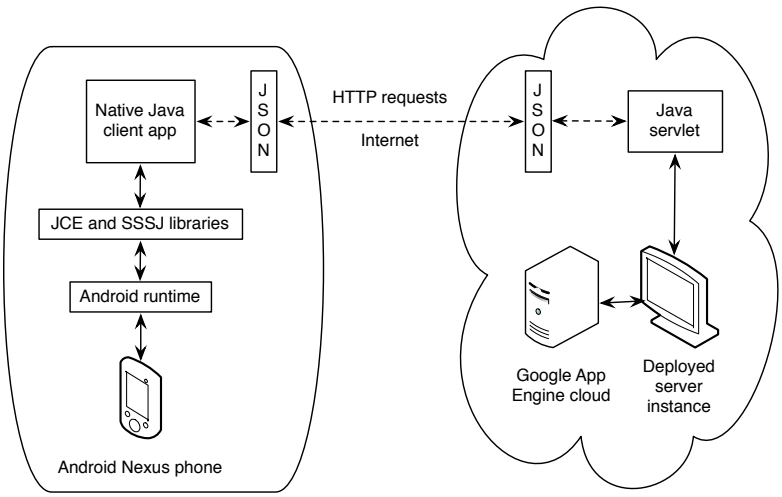

Figure 2: High-level model of implementation.

The simulation consisted of multiple iterations of encryption and decryption using the proposed functions defined in Section 4, using a single-attribute policy to configure an environment that ran at the fastest possible speed. A "Type A" pairing was utilized in the algorithm, as specified in the jPBC library source. Performance benchmark results are shown in Table 7.1, showing the average execution times of all main cryptographic operations calculated from 100 simulation runs on the same desktop computer platform to permit direct comparison. Simulations from the original BSW algorithm [2] as well as the proposed algorithm are shown; in the latter case, one run was made using an optional group key to additionally secure the plaintext, with a round of re-encryption included, and one run was made without the benefit of a group key.

Cryptographic function	**Baseline (BSW)**	**Proposed (no GK)**	**Proposed (with GK)**
SETUP by data owner.	46	22	31
SETUP by manager.	n/a	18	17
KEYGEN.	68	70	66
ENCRYPT by data owner.	59	61	57
ENCRYPT by CSP.	n/a	n/a	17
DECRYPT.	23	23	40
RE-ENCRYPT setup.	n/a	n/a	20
RE-ENCRYPT.	n/a	n/a	7

Table 4: Performance benchmarks (in ms) on an iMac desktop computer.

Next, operations were repeated on the appropriate platform (desktop, mobile, or cloud instance) for each operation, to ascertain realistic timings in a mobile cloud computing system. A user interface was built for the Android app to allow execution of all algorithms locally or on a Google App Engine instance in the cloud, as depicted in Figure 3. The results are summarized in Table 7.1, with the appropriate platform chosen to represent a typical device executing each operation in question. Note that the underlying jPBC library is not optimized for a constrained mobile operating environment; such optimizations may often yield very significant performance improvements in practice. For instance, careful memory allocation and maintenance of a small footprint may reduce the expensive garbage collection events observed on the Android devices during execution.

Cryptographic function	**Device**	**Baseline (BSW)**	**Proposed (no GK)**	**Proposed (with GK)**
SETUP by data owner.	Note II	1157	396	586
SETUP by manager.	GAE	n/a	278	219
KEYGEN.	GAE	791	749	786
ENCRYPT by data owner.	Note II	1273	1250	1239
ENCRYPT by CSP.	Note II	n/a	n/a	657
DECRYPT.	Note II	1364	1391	2043
RE-ENCRYPT setup.	GAE	n/a	n/a	247
RE-ENCRYPT.	GAE	n/a	n/a	130

Table 5: Performance benchmarks (in ms), with "Note II" denoting the Galaxy Note II mobile phone, and "GAE" denoting an F1 instance running on a Google App Engine cloud servlet.

The comparative benchmarking results are shown visually in Figure 4. The following observations may be made:

- In comparison to the baseline implementation, the key setup activity in the proposed protocol

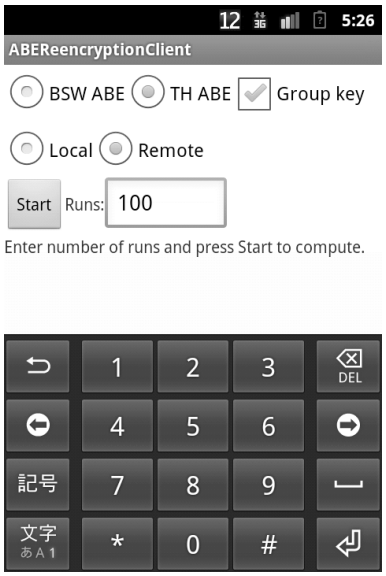

Figure 3: User interface on the Android app used for benchmarking, running on the Nexus One and Galaxy Note II smartphones.

is approximately evenly split between the data owner and the manager, which is of high benefit given that the owner is presumed to be a resource-constrained mobile user.

- The key generation and encryption activities are approximately equal in terms of computational requirements. The utilization of a group key only applies an approximately 30% penalty to encryption, which is borne by the CSP because it is responsible for the pairing operation, not the data owner.
- Crucially, the data owner does not participate in the key generation activity in the proposed protocol; the manager does so instead.
- The optional re-encryption operation is only a fraction, approximately 27%, of the total encryption operation in terms of the period of computation, and is also performed by the CSP without burdening the data owner.

7.2 Simulation

A custom simulation program was developed which permits an assessment of the scalability of the proposed scheme. The simulation program was executed on a desktop computer but was calibrated with the function timing results from the benchmarks obtained as described in the previous Section 7.1. Various parameters may be adjusted in the simulation which highlight the differences in the algorithms discussed.

An initial unauthorized user population is modelled, and in each round of the simulation, users randomly join or leave a user set that is authorized to access a particular data record. A single data owner responsible for data encryption is modelled. Each user randomly takes an action each round, with some predefined probability; actions include accessing the encrypted data and performing a decryption, or joining or leaving the authorized user set and thus triggering appropriate key generation activities. The encrypted data record stored in the cloud may also be replaced in a

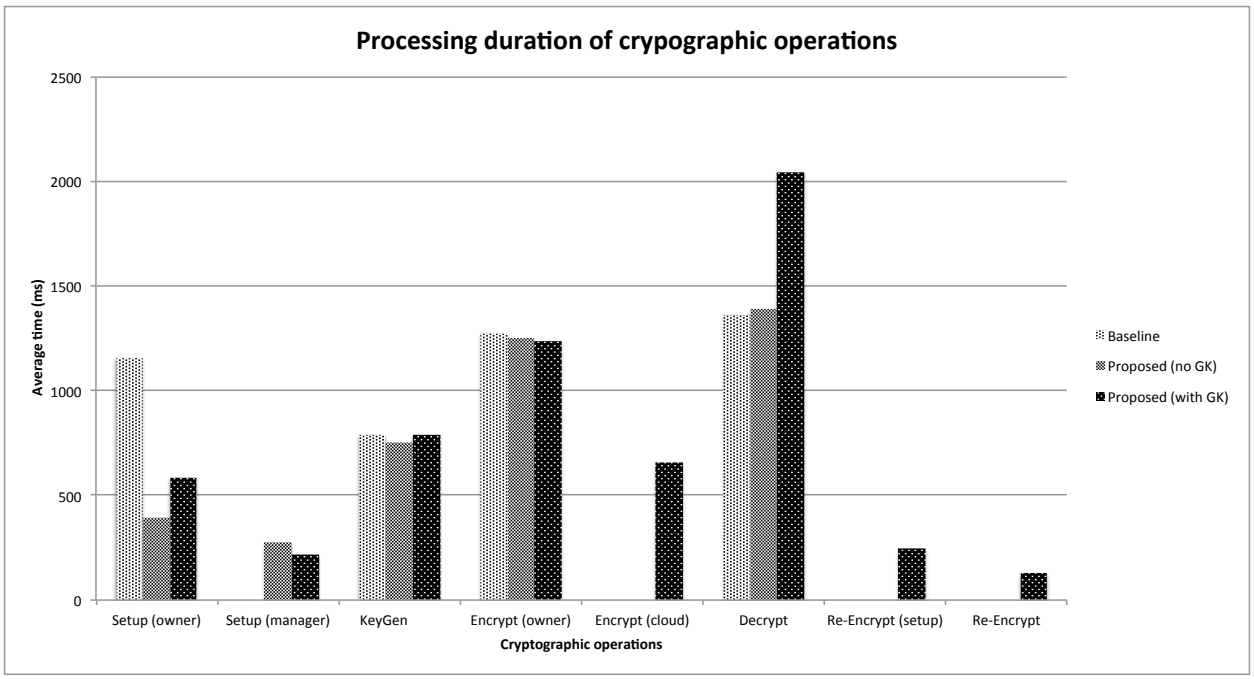

Figure 4: Processing time of cryptographic operations in the baseline and proposed protocols.

round by the data owner, once it has outlived its usefulness, with more recent data; this initiates a new key setup phase.

In Figures 5, 6, 7, and 8, the simulation results for one simulation run of each algorithm are shown, with the processing workload shown over time for each entity (the data owner, the manager, the CSP, and the total set of users involved in accessing the data record stored in the cloud). The workloads are directly based on the cryptographic function profiling results found in Section 7.1 so that calibration was done with real-world data. The simulation was run with the adjustable parameters specified in Table 7.2. The irregularities found in the plots are due to the probabilistic nature of the events executed in the simulation.

The following observations may be made with respect to the results of the illustrated runs showing various dominant roles in the system:

Parameter	**Value**
Initial unauthorized user population	10,000 users
Length of each round	1 hour
Total length of all rounds simulated	1 year
Probability of a user joining the authorized set	0.5%
Probability of a user leaving the authorized set	0.5%
Probability of a user downloading the cloud data	5%
Probability of the cloud data being replaced	5%
Total joins in simulation run	397,000
Total accesses in simulation run	396,000
Total leaves in simulation run	40,000
Total data replacements in simulation run	419

Table 6: Parameters for simulation.

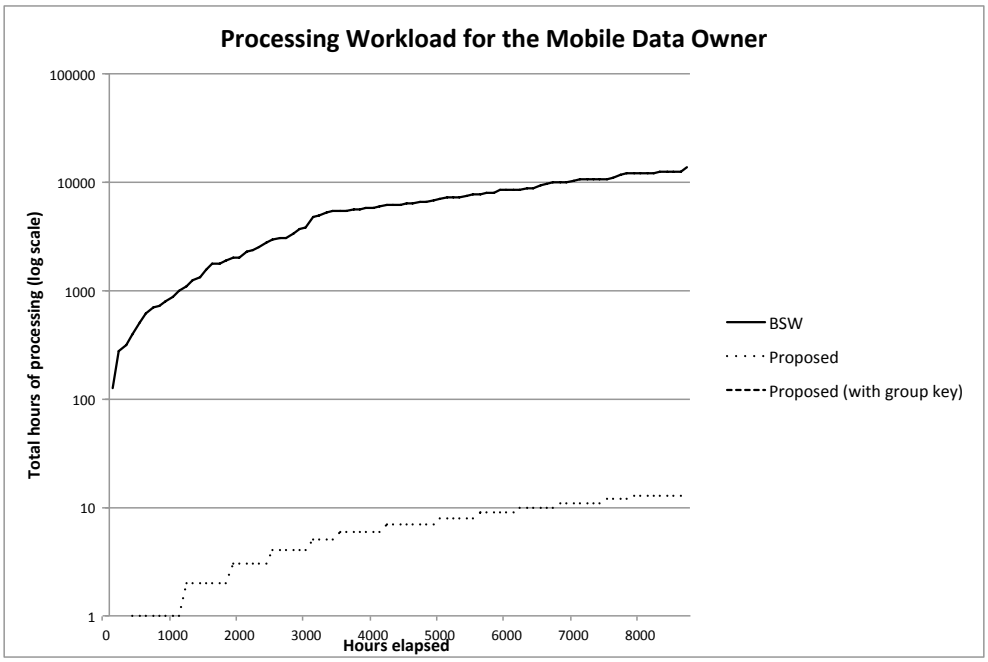

Figure 5: Processing workload for the mobile data owner.

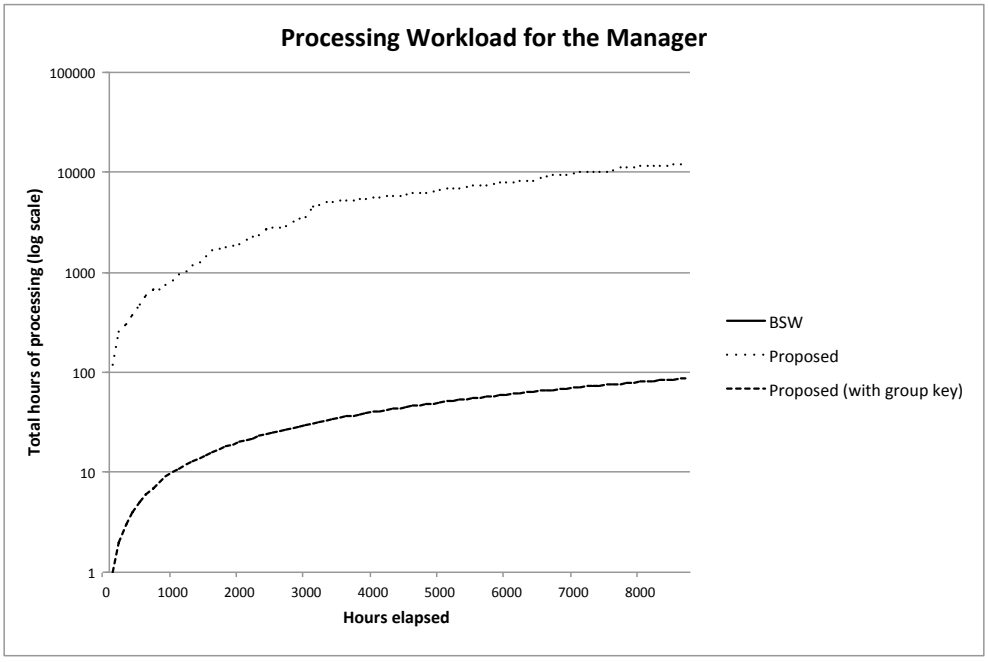

Figure 6: Processing workload for the manager.

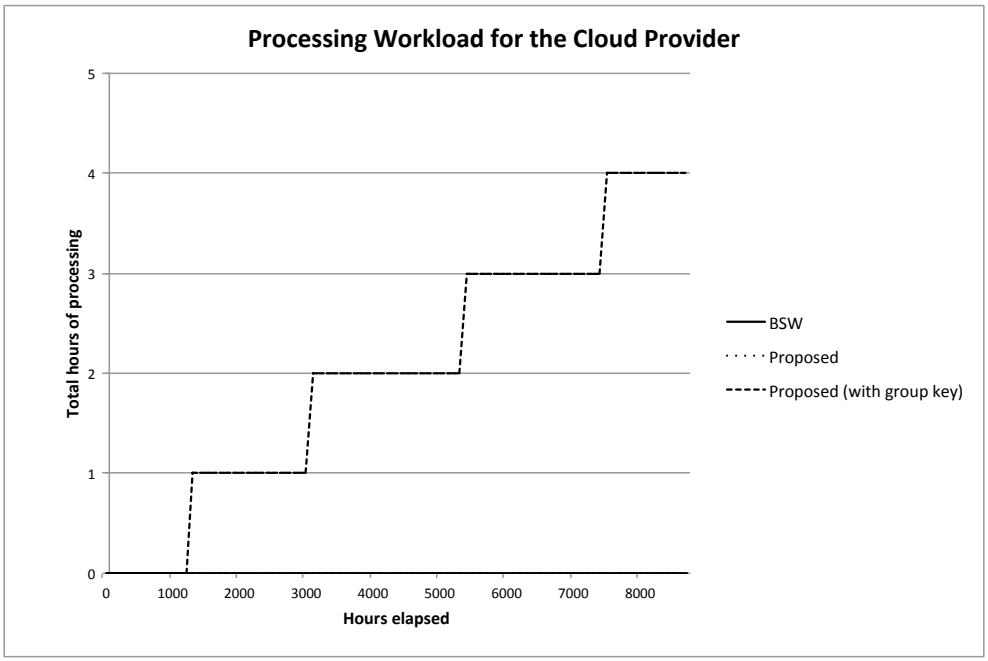

Figure 7: Processing workload for the cloud provider.

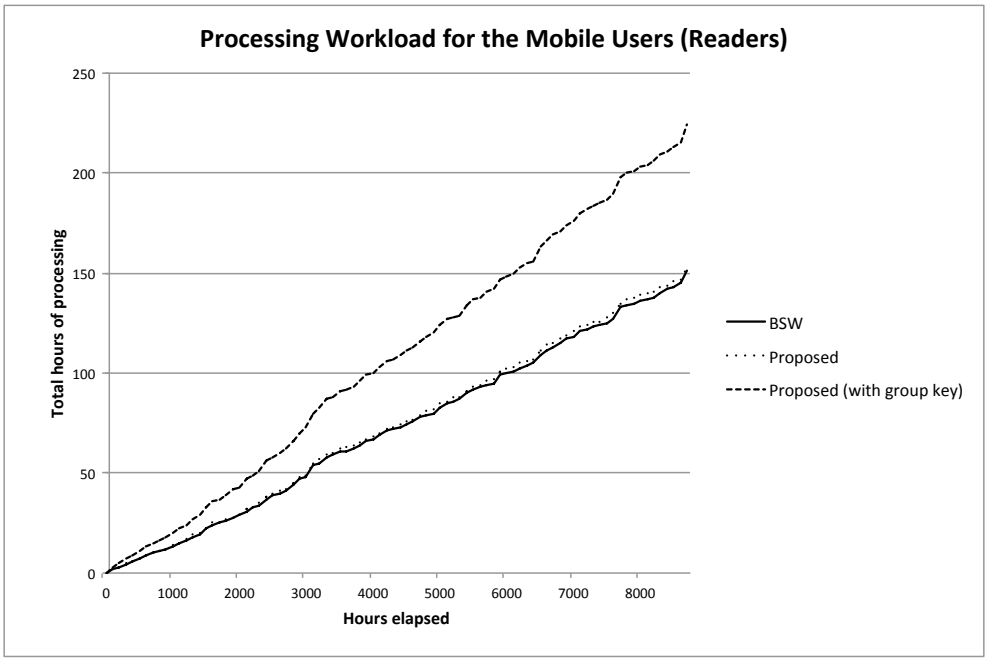

Figure 8: Processing workload for the mobile user (reader) population.

1. In the BSW algorithm [2], the dominant workload is undertaken by the data owner, which participates in not only the encryption of the user data, but also in the data secret key generation for each new user, as shown in Figure 5. The owner must also re-generate keys for all users whenever a revocation occurs, without assistance of any other network entity, based on the assumption that revocation is only possible through modification of attributes for the user in question. Since the data owner is presumed to be a mobile device in the assumed system model, the scalability potential appears inadequate.

2. In the proposed algorithm without the use of a group key, the manager becomes responsible for the main workload of the key re-generation activity, which entails a pairing operation, as shown in Figure 6. The manager is expected to be able to scale accordingly to meet the processing demands, but requires sufficient client infrastructure to do so, which may be uneconomical.

3. In the proposed algorithm with the use of a group key, the manager is still responsible for most of the key re-generation activity, but revocation is now handled through re-encryption of the group key, a task performed by the cloud provider, as shown in Figure 7; this results in a much lower overall workload in the system. The additional decryption cost for the reader population is also modest, as shown in Figure 8. Hence, this algorithm is considered the best candidate for a highly scalable mobile cloud computing application.

8 Conclusions

A key management system has been proposed for data outsourcing applications, whereby attribute-based encryption effectively permits authorized users to access secure content in the cloud based on the satisfaction of an attribute-based policy. The scheme has been modified so that a data owner and a trusted authority co-operate in the key generation and encryption processes such that computationally-intensive cryptographic operations and requests are minimized for the data owner; this is of importance to a population of mobile users that must conserve their consumption of battery and usage of wireless communication. Furthermore, a hybrid protocol is proposed that optionally allows message encryption based on a group key, allowing the user membership to be further refined. Additionally, it allows re-encryption to occur, and thus revocation to become efficient without necessitating existing common remedies and their limitations, such as expiration of attributes specified in the attribute-based policy. The proposed protocol in similar in overall performance to the original ciphertext-policy attribute-based-encryption idea, while significantly lessening the computational and traffic burden on the mobile data owner. Thus, the proposal is useful for securing mobile cloud computing with very large user populations.

Acknowledgments

This work was supported in part by a National Sciences and Engineering Research Council (NSERC) grant awarded to Dr. Hasan, and an NSERC CGS Doctoral scholarship awarded to Piotr Tysowski.

References

[1] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 280–293.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, ser. SP '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 321–334.

[3] A. Tassanaviboon and G. Gong, "OAuth and ABE based authorization in semi-trusted cloud computing: aauth," in *Proceedings of the second international workshop on Data intensive computing in the clouds*, ser. DataCloud-SC '11. New York, NY, USA: ACM, 2011, pp. 41–50.

[4] X. Liang, R. Lu, and X. Lin, "Ciphertext policy attribute based encryption with efficient revocation," University of Waterloo, Technical Report BBCR, 2011.

[5] J. Hur and D. K. Noh, "Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 1214–1221, 2011.

[6] P. Tysowski and M. A. Hasan, "Towards Secure Communication for Highly Scalable Mobile Applications in Cloud Computing Systems," Centre for Applied Cryptographic Research (CACR), University of Waterloo, Tech. Rep. 33, 2011.

[7] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions of Information and System Security*, vol. 9, pp. 1–30, Feb. 2006.

[8] Q. Liu, G. Wang, and J. Wu, "Clock-based proxy re-encryption scheme in unreliable clouds," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, sept. 2012, pp. 304 –305.

[9] J.-M. Do, Y.-J. Song, and N. Park, "Attribute based proxy re-encryption for data confidentiality in cloud computing environments," in *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, may 2011, pp. 248 –251.

[10] J. Wang, "Java Realization for Ciphertext-Policy Attribute-Based Encryption," 2012. [Online]. Available: http://github.com/wakemecn

[11] A. De Caro, "Java Pairing-Based Cryptography Library," 2012. [Online]. Available: http://libeccio.dia.unisa.it/projects/jpbc/

[12] B. Lynn, "PBC (Pairing-Based Cryptography) Library," 2012. [Online]. Available: http://crypto.stanford.edu/pbc/