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

Piotr K. Tysowski and M. Anwarul Hasan, *Senior Member*, *IEEE*

Abstract—Outsourcing data to the cloud are 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 are 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 that 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 realistic workload in a mobile cloud computing system.

Index Terms—Distributed computing, mobile computing, security, cryptography, scalability

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, are 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 carry the persistent risk of being intercepted by an unauthorized party despite safeguards promised by the provider. Therefore, it is useful to apply software techniques, such as encryption key management, 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 tasks.

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 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

• *The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada.*

Manuscript received 20 Feb. 2013; revised 19 Aug. 2013; accepted 23 Oct. 2013; published online 31 Oct. 2013.
Recommended for acceptance by M. Demirbas.
For information on obtaining reprints of this article, please send e-mail to: tcc@computer.org, and reference IEEECS Log Number TCC-2013-02-0033.
Digital Object Identifier no. 10.1109/TCC.2013.11.

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 and messaging burdens.
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.
5. The real-world performance of the proposed protocol is demonstrated on commercially popular mobile and cloud platforms. In addition, simulations show the scalability of the protocol in terms of computational workloads within a very large mobile user population.

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 in terms of its security and performance 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. Section 8 provides concluding remarks. An earlier version of this work appears as a technical report [1].

2 Related Work on Proxy Re-Encryption and Attribute-Based Encryption

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 such as RSA [2] (which stands for the authors Rivest, Shamir, and Adleman and depends on the difficulty of factoring large integers) is that it requires that the data owner provide an encrypted version of data for each recipient that may access it. If user data are 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 redistribution through additional communication sessions to handle user revocation; in a highly scalable system, such events may occur at relatively high frequency. Wireless communication, however, is expensive and results in rapid battery drain, especially when transmitting [3].

Data should ideally be stored in the cloud in encrypted form so that the cloud provider cannot access it. This notion is dependent on the keys being securely managed by an entity outside of the provider's domain. The difficulty arises when new users join the system, and existing ones leave, necessitating new keys to be generated. The encrypted data should ideally be transformed such that it may be unlocked with new keys, without an intermediate decryption step that would allow the cloud provider to read the plaintext; this process is known as *data re-encryption*. Although it appears to be a promising technique in managing encrypted data as access rights evolve over time, current solutions in the literature do not address the issue of high scalability to a sufficient and satisfactory degree; nor do they necessarily strive to lessen the computational and communication burden on users connecting to the cloud from resource-constrained mobile devices.

The technique of ciphertext-policy attribute-based encryption (CP-ABE) [4] offers numerous advantages. 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 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. The owner may not necessarily be required in every read transaction. Normally, a scheme based on CP-ABE relies upon the data owner granting access permission through an access tree, which requires his or her constant availability. In some works, 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 [5]; the solution, however, is not tailored for a mobile environment due to its computational demands, the required constant availability of the data owner, and time-based expiration of access leading to frequent key retrieval.

Revocation of an authorized user is particularly hard to accomplish efficiently in CP-ABE and is usually addressed by extending attributes or keys with expiration dates. A tree of revocable attributes may need to be maintained and a trusted party assigned to validate revocation status. A mechanism using linear secret sharing and binary tree techniques is one example [6], but mobile users have to incur the communication cost of continually requesting new keys. Also, the data owner is typically also a mobile user, and thus cannot manage access control on demand for all due to its transient connectivity. Revocation has been proposed that relies on stateless key distribution and access

control on the attribute level [7], but requires a trusted authority and encumbers the owner with a cryptographic pairing operation that is computationally costly.

Another related approach combines Hierarchical Identity-Based Encryption (HIBE) and CP-ABE [23], using hierarchical domain masters to distribute user keys; this is done at the cost of increased storage requirements for key material held by users and a greater amount of processing when generating ciphertext. A method of trusted data sharing has been proposed that uses a progressive elliptic curve encryption scheme [8]. However, it relies upon a writer uploading encrypted data to the cloud, then distributing credentials to the cloud to perform re-encryption, and also to the reader on each data access attempt; this is clearly impractical when applied to resource-constrained devices and networks. 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 [9]; its limitation is that it requires granting access solely based on user identity.

Various other proxy re-encryption schemes have been applied to secure distributed storage. One method is to re-encrypt the stored content during retrieval. Such a technique has been applied to an encrypted file storage system where a content owner encrypts blocks of content with unique, symmetric content keys, and these keys are then further encrypted to form a *lockbox* [10]; users communicate with an access control server to decrypt them. The problem is that the content owner manages access control for all other users, which is a great burden, and requires dynamic re-encryption of the same data whenever multiple users access it. In the model herein, one-time re-encryption only occurs whenever membership changes, presumably a less frequent occurrence than that of data access. Other approaches require a trusted proxy for each decryption [11], which increases the communication cost.

Proxy re-encryption has also been combined with CP-ABE [12] such that re-encryption keys are computed by the cloud provider based on a secret that is preshared 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 [13]; 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. A different approach has been suggested where attribute revocation events occur, and in response, an authority redefines master key components for the attributes, user secret keys are updated to a new version, and data are re-encrypted by the proxy server [14]; the difficulty is that revocation is dependent upon modifications to attributes, resulting in costly key updates on each revocation. Finally, a technique that combines CP-ABE with proxy re-encryption [15] does not appear to be highly efficient for mobile users: for instance, the decryption process requires processing two access subtrees instead of one.

Another related work proposes the merging of ABE with proxy re-encryption, allowing fine-grained access control of resources while offloading re-encryption activity to the cloud provider [16]. It has numerous differences to the scheme that will be proposed. The data owner is involved in generating a key for each new user that joins or leaves the system, rather than offloading this task; it is not only a prohibitive cost for a mobile user, but also impractical due to the user's mobility. Another difference is that a secret key must be regenerated and redistributed for each user, in lazy fashion, whenever user revocation occurs, rather than allowing users to upgrade a common group key, which reduces the communication cost and results in higher efficiency. Furthermore, the re-encryption occurs due to attribute redefinition and the scheme is based on key-policy attribute-based encryption (KP-ABE) and not CP-ABE, where the ciphertext is associated with a policy.

A multiauthority system has been proposed [17] that uses attribute-based access control to avoid the single point of failure of a single key authority, and relies upon the data owner communicating with attribute authorities to generate multiple encryption keys for hosted content. The costs of communicating with multiple authorities and storing multiple keys could become prohibitive for mobile users. In a related work [18], each user submits multiple secret keys issued by authorities to a server to generate a decryption token for each ciphertext that is used in concert with the user's global secret key to perform a decryption. The same costs associated with multiple authorities apply, however, and such authorities complicate and add to the expense of system engineering.

The authors are unaware of a similar scheme where fine-grained operations in attribute-based cryptography have been reassigned across system components to minimize the workload of mobile users; nor are other techniques found with performance benchmarked on commercial mobile and cloud systems, as herein, useful in assessing real-world viability.

3 Model of Mobile Cloud Computing

3.1 System Model

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

Requests are made over the public Internet, which is considered a normally reliable but insecure medium. A network access model is shown in Fig. 1. Internet traffic is routed through a topology of network switches, which culminate in individual Internet service provider (ISP) network switches connected by high-capacity optical links such as OC-3; this packet data network may be bridged to a wireless 3G or 4G infrastructure through a gateway node, allowing smartphones to connect wirelessly to 3G or 4G towers and switches. Additionally, smartphones may communicate among themselves using short-range local links such as Bluetooth. Another entry point into the Internet is via a router and Wi-Fi access point, enabling notebooks and wireless sensors to connect via some Wi-Fi

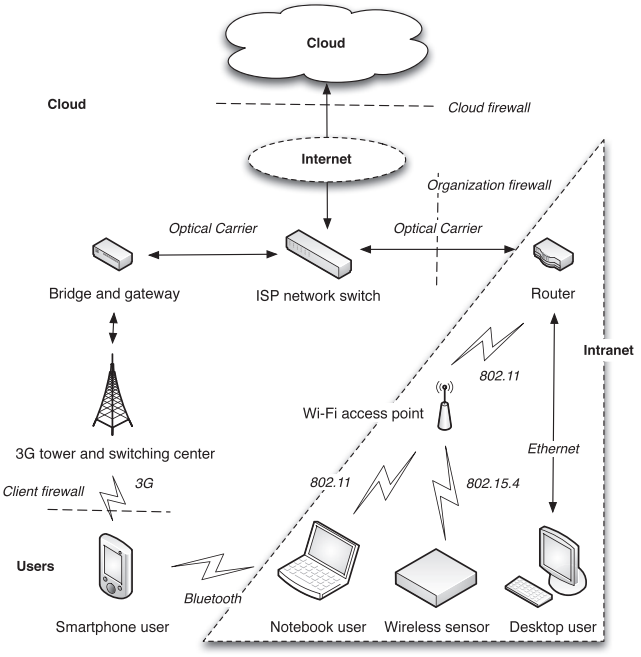


Fig. 1. Cloud computing network access model.

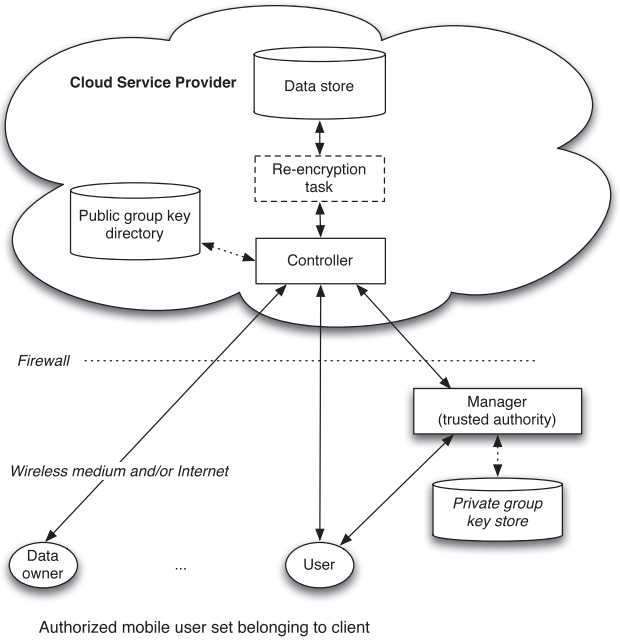


Fig. 2. Cloud computing system model.

standard, such as 802.11 or low-cost 802.15.4 as part of a personal area network.

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 mobility and limited wireless network coverage areas, mobile users will likely suffer from transient connectivity such that their constant availability in the system cannot be guaranteed. Representative recent-generation specifications of in-market devices are listed in Table 1.

The system model of the proposed work is shown in Fig. 2. 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 is considered fast and scalable, but less capable than the public cloud. 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 limited battery lifetime 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 networks.

TABLE 1
Example Specifications for a Smartphone

Criterion	Specification
Processor	Single/dual-core, 0.6 to 1 GHz
Memory	256 to 512 MB
Battery capacity	1200 mAh
Power	Up to 1200 mAh per day
Wireless data rate	2800 kbps download (3G)

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.

TABLE 2
A Legend for the Symbolic Notation Used

Symbol	Description
CSP	Cloud service provider.
M	Trusted manager.
T	Access tree structure.
R	Root node of T.
A	Set of attributes to satisfy 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	Group public key of U_r.
GSK	Group secret key of U_r.
$DDSK$	Delegated data secret key.
$RK_{0\to x}$	Re-encryption key from version 0 to x.

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 public 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 scalable manner for mobile cloud computing applications. Table 2 summarizes the symbolic notation used throughout the description. In the discussion that follows, the following improvements are proposed to the basic functions of the original CP-ABE scheme [4], such that key components have been reassigned to the various entities in the system model to achieve scalable key management while reducing the mobile user computational and communication workload:

- A single authority does not generate all key material; the mobile data owner and cloud entity cooperate 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 possesses highly scalable computational ability, unlike a resource-constrained mobile user; a trusted manager also has greater computational resources. Pairing operations, which are the most expensive cryptographic operations involved in the proposed protocol, are thus performed by the cloud or manager to the maximum extent, relieving the burden of 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, thus, a hybrid approach, combining cryptographic techniques, 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: $\forall g_0, g_1 \in G_0 : e(g_0^\alpha, g_1^\beta) = e(g_0, g_1)^{\alpha\beta}$.
- Nondegeneracy: $e(g_0, g_1) \neq 1$.
- Computability: $\forall g_0, g_1 \in G_0$, there is an efficient algorithm to compute $e(g_0, g_1)$.

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

$Setup() \to PPK, PSK, OSK$: 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 are 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 precomputed 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:

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

To provide an additional layer of security, the manager may create a group secret key GSK; this key is a shared secret for an individual user or a restricted subset of users U_r, equal to a random value $u_0 \in \mathbb{Z}_p^*$. The group public key GPK is derived as follows:

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

This GPK is uploaded to the public directory, as well. The secret key GSK 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 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 cloud's public directory, 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 [4]. 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.

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 the GPK 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 pairing operation $e(g,g)^{u_0}$ may be performed by the cloud or manager to assist the data owner.

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:

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

Re-Encrypt$(CT_0, RK_{0 \rightarrow x}) \rightarrow CT_x$: 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 immediately, or distributed to each user on demand through a secure offline 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 $RK_{0 \rightarrow x}$ from a user holding secret group key GSK_x assigned to version x; or, $RK_{0 \rightarrow x}$ 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:

$$RK_{0 \rightarrow x} = \left\{ g^{\frac{u_x}{u_0}} \right\}.$$

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 (2):

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

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 CSP retains the components $C_{0_{msg}}$ and $C_{x_{base}}$ 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 that 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 each other. The data owner is not involved in the key generation and need not remain available at that time.

To generate the additional required subparts 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, without requiring the participation of the data owner. 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.

Decrypt$(CT, DSK, PPK, GSK) \rightarrow m$: Any user that is authorized, by virtue of holding the required attributes A, 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

TABLE 3
A Summary of the Key Material and Activities for Each System Entity

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 the 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)	Publishes PPK and GPK in a public directory. Optionally performs a pairing 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.

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 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 (3), and C_x and C_x' derived from CT_{own}, as found in (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 nonleaf node, is described in detail in [4]. 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:

$$\begin{aligned}\text{DECRYPTNODE}(CT_{own}, DSK, R) &= e(g,g)^{rq_R(0)}\\ &= e(g,g)^{rs}.\end{aligned}$$

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 component $g^{\frac{1}{\beta}}$ in the PPK to compute the decryption component $g^{\frac{u_x}{\beta}}$.

The message m can then be decrypted as follows, assuming one round of re-encryption:

$$\begin{aligned}m &= \frac{\tilde{C}_x \cdot e(g,g)^{rs}}{e(g^{\beta s}, D) \cdot e(g^{\beta s}, g^{\frac{u_x}{\beta}})}\\ &= \frac{m \cdot e(g,g)^{\alpha s + u_x s} \cdot e(g,g)^{rs}}{e(g^{\beta s}, g^{\frac{(\alpha+r)}{\beta}}) \cdot e(g^{\beta s}, g^{\frac{u_x}{\beta}})}\\ &= \frac{m \cdot e(g,g)^{(\alpha+r+u_x)s}}{e(g,g)^{(\alpha+r+u_x)s}}.\end{aligned}$$

Note that the component $e(g^{\beta s}, g^{\frac{(\alpha+r)}{\beta}})$ in the above equation is precomputed 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 3. The cryptographic operations described in this section, applied to a typical encryption and decryption transaction, are summarized in Table 4.

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 data secret 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:

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

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

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

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

6.1 Security Analysis

The proposed scheme offers a dual layer of security through the combination of attribute-based encryption and public key encryption techniques, the latter which may be optionally applied. The security proof for the underlying attribute-based cryptography appears in [4] and is relied upon here. The security intuition of the proposed scheme is now described.

TABLE 4
A Summary of Operations, with Participating Actors Shown, Assuming No Additional Re-Encryption Occurs

	Alice, Data Owner (U_o)	**Cloud** (CSP)	**Manager** (M)	**Bob, Recipient** ($\in U_r$)
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 CSP, respectively.	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.	May assist the encryptor with generation of ciphertext CT_{own}. 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.

In the basic case, where only attribute-based encryption is utilized, the data owner generates the owner secret key OSK and does not share it. The public form of the key, the owner public key OPK, is used to generate the data secret key DSK necessary to decrypt the data. The data secret key is unique to each collection of attributes sufficient to decrypt the data. Thus, the security of the system relies upon the trustworthiness of the data owner in keeping the owner key secret; the owner has the freedom of sharing its data with any user directly, anyway. If a user lacking the required attributes attempts to decrypt the ciphertext, then such an attempt will be unsuccessful, as the wrong access tree will be input. Note that the requisite set of attributes to perform decryption may be shared by multiple users who are all equally privileged.

Any user may encrypt data by using the public partition key PPK stored in the cloud and by expressing an access policy based on particular attributes. 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; the former is retained by the data owner, and the latter kept by each intended recipient. The algorithm achieves collusion resistance because the $e(g,g)^{\alpha s}$ term of the ciphertext $C_{0_{msg}}$ cannot be recovered by an attacker even if the manager's or a user's private keys are compromised. In this case, the $C' = g^{\beta s}$ term in CT_{own} in (1) and $D = g^{\frac{(\alpha+r)}{\beta}}$ in DSK in (3), when combined, results in *blinding* by the randomized value $e(g,g)^{rs}$ as explained in [4], which can only be removed by possession of the correct attributes in the execution of the DECRYPT operation.

Furthermore, 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 group public key GPK. Only the users possessing the group secret key GSK will be allowed access to the stored ciphertext, as an additional security measure. If the secret group key GSK held by the authorized user set is compromised, the data are still safeguarded to a significant extent; only the users that have the required attributes will be able to decrypt it, as a fallback. Furthermore, the data owner can request proxy re-encryption by the cloud provider, which has the effect of making the compromised version of the GSK obsolete.

At all times, the interception of any key components over the network will not yield useful information to the attacker, as no private keys are transmitted in the clear. For instance, the computation of the DSK by the manager relies upon a component in the published partition public key PPK, and not the secret key held by the data owner. Nor is useful information revealed to the CSP during the re-encryption process. From the re-encryption key $RK_{0\to x} = g^{\frac{u_x}{u_0}}$ provided by the data owner or manager, the group secret key $GSK = u_0$ cannot be feasibly computed by the CSP, although it will use the re-encryption key to carry out the actual re-encryption operation.

6.2 Performance Analysis

Critically, performance implications are modest for 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. Also, with the assistance of the manager, the user performs only a single pairing operation on decryption.

The group key may be shared among a group or simply possessed by a single user; the tradeoff made in its use 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. The manager can also assist with distribution. Furthermore, 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 [9].

7 IMPLEMENTATION

The proposed protocol was implemented and profiled to gauge its performance. It was realized on popular existing

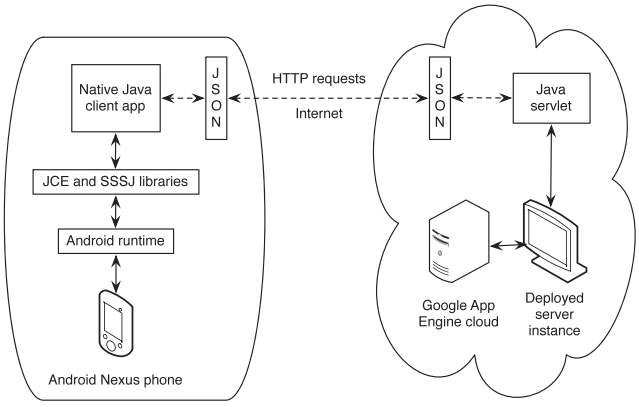


Fig. 3. A high-level model of the implementation.

commercial platforms, including the Google Android mobile and the Google App Engine (GAE) 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 in Java [19], [20] that relies upon the original CP-ABE scheme [4] 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) version 1.2.1 [21], a port of the pairing-based cryptography library (PBC) in C [22]. The use of Java 6 Standard Edition permits the protocol to be ported to a wide range of computing environments. Although a version of the implementation in C or C++ was not tested, the Java-based jPBC library has been found to have comparable performance to the C-based PBC library [20]. An implementation in Java is useful to evaluate because of its development kit support on popular mobile platforms such as Android and BlackBerry, and cloud platforms such as the Google App Engine and Amazon web services.

The implementation was run on different computing hosts to assess their relative performance. Refer to the implementation model in Fig. 3. 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 single-core 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 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 JavaScript Object Notation (JSON) for data interchange and the Google Gson library for marshaling 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 is whitelisted on Android and GAE; fortunately, the required Java Cryptography Extension (JCE) classes are supported.

TABLE 5
The Performance Results Obtained from the Implementation Running on an iMac Desktop Computer

Algorithm	Task	Timings (in ms)			
		μ	σ	LB	UB
Baseline (BSW)	Owner setup	47.5	28.1	0	102.5
	Keygen	70.0	3.7	62.8	77.0.2
	Owner encryption	61.3	2.9	55.7	66.9
	Decryption	23.9	1.5	20.9	26.8
Proposed (no group key)	Owner setup	29.8	45.3	0	118.6
	Manager setup	19.8	2.1	15.7	23.8
	Keygen	74.6	5.4	64.1	85.1
	Owner encryption	65.0	4.9	55.3	74.6
	Decryption	24.5	2.5	19.6	29.3
Proposed (with group key)	Owner setup	37.6	26.7	0	89.8
	Manager setup	18.8	1.2	16.4	21.2
	Keygen	70.0	3.6	63.0	77.0
	Owner encryption	60.8	2.3	56.3	65.2
	Cloud encryption	18.6	1.0	16.6	20.5
	Decryption	42.8	2.1	38.6	47.0
	Reencryption setup	21.6	1.0	19.7	23.5
	Reencryption	7.5	0.6	6.2	8.8

The following symbols are used: μ is the sample mean, σ is the sample standard deviation, and LB and UB are the lower and upper bounds, respectively, of the 95 percent confidence interval, for each set of runs using each of the baseline and proposed algorithms. Hundred runs were executed using each technique, and all operations were performed using a single-attribute policy. The data owner performed a pairing operation on encryption in all cases, without assistance, which accounts for its slower operation.

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 with a group order size of 160 bits and a base field order size of 512 bits, which is the default curve configuration in the jPBC library test code, and is suitable for cryptographic use. Performance benchmark results are shown in Table 5, showing the average execution times of all main cryptographic operations calculated from simulation runs on the same iMac desktop computer platform to permit direct comparison. Simulations from the original BSW algorithm [4] as well as the proposed algorithms are shown; in the latter case, one set of runs was made using an optional group key to additionally secure the plaintext, with a round of re-encryption included, and one set of runs was made without the benefit of a group key, to permit comparison.

Next, operations were repeated on the appropriate platform (desktop, mobile, and cloud) 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 Fig. 4. The results are summarized in Table 6, with the appropriate platform chosen to represent a typical device executing each operation in question. The same curve parameters were used as described above.

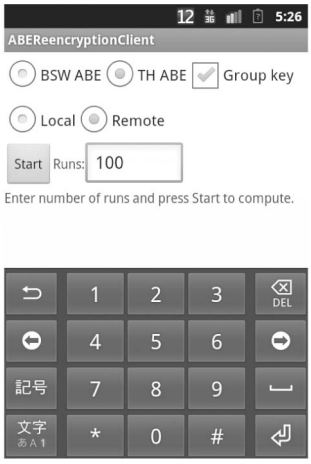


Fig. 4. The user interface of the mobile client application.

Timings from the Galaxy Note II smartphone were adopted, since it is two-and-a-half years newer than the Nexus One smartphone and more representative of current-generation high-end models. However, as a point of comparison between a quad-core and single-core phone, the Nexus One smartphone was approximately 44 percent slower in setup, 51 percent slower in encryption, but 32 percent faster in decryption tasks than the Note II.

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 maintaining a small footprint may reduce the expensive garbage collection events observed in the Android device system logs during execution. The comparative benchmarking results are shown visually in Fig. 5, with the following observations:

- In comparison to the baseline implementation, the key setup activity in the proposed protocol 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 about a 30 percent penalty to encryption, which is borne by the CSP because it performs the pairing operation, not the data owner.
- Crucially, the mobile data owner does not participate in the data secret key generation activity in the proposed protocol; the manager does so instead.
- Furthermore, the data owner is not required to perform costly pairing operations, including in the generation of ciphertext (although it does so in the implementation).
- The optional re-encryption operation is only a fraction, approximately 40 percent, of the total encryption operation in terms of the period of computation, and is also performed by the CSP without burdening the data owner. Advantageously, the CSP has the potential to be scaled by allocating additional cloud instances as required.

TABLE 6
The Performance Benchmarks Used for Calibration of the Simulation

Crypto. 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.	GAE	n/a	n/a	272
DECRYPT.	Note II	1364	1391	2043
RE-ENCRYPT setup.	GAE	n/a	n/a	247
RE-ENCRYPT.	GAE	n/a	n/a	130

All timings are in ms, with "Note II" denoting the Galaxy Note II mobile phone, and "GAE" denoting an F1 instance running on a GAE cloud servlet.

7.2 Simulation

A custom simulation program was developed that permits an assessment of the scalability potential of the proposed scheme. The 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; that is, the timings served as the basis for calculating the accumulated processing workload of the various system

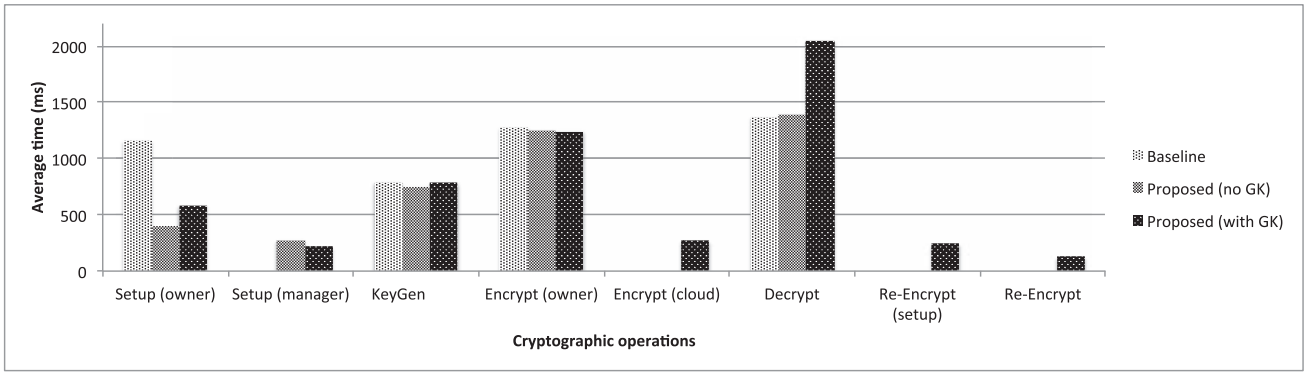


Fig. 5. The performance results obtained from the implementation showing the processing time of cryptographic operations in the baseline and proposed protocols on various appropriate platforms, as derived from Table 6.

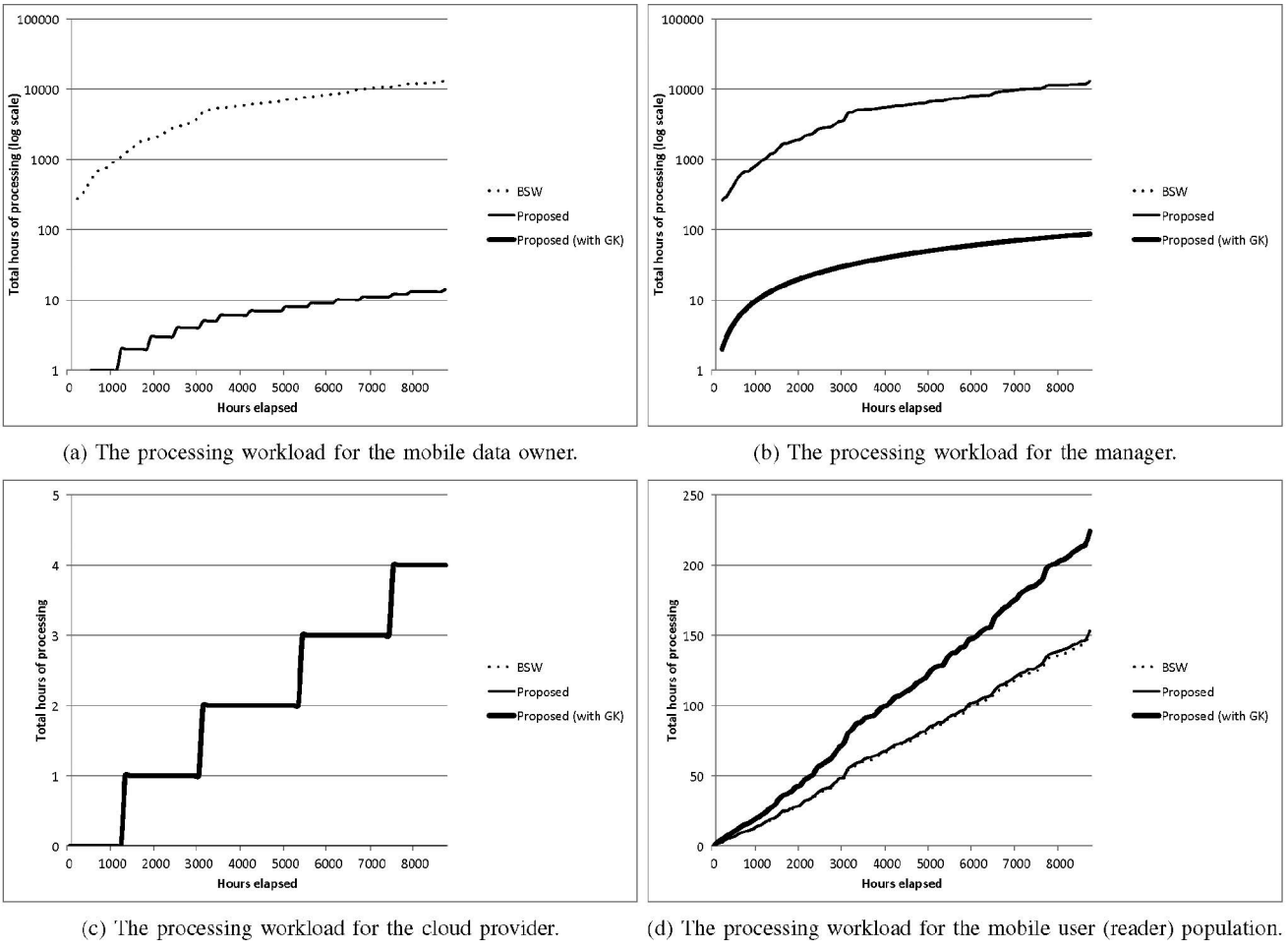


Fig. 6. The processing workloads, for all the system entities, attained from the simulation.

entities as a result of performing the defined cryptographic operations in response to simulated user actions. The number of communication sessions was also tracked. Various parameters may be adjusted in the program to highlight the differences in the algorithms, including the original CP-ABE and the proposed schemes (with or without a group key and ciphertext re-encryption).

An initial unauthorized user population is modeled, 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 modeled. 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 round by the data owner, once it has outlived its usefulness, with more recent data; this initiates a new key setup phase.

In Fig. 6, the simulation results for one typical 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 obtained from the practical implementation. The simulation was run with values for adjustable parameters as specified in Table 7. The irregularities found in the plots are due to the probabilistic nature of the events executed in one sample simulation execution. The difference between the total joins and leaves is the result of the occasional simulated updating of cloud data by the data owner, causing immediate deauthorization of the entire user population, which leads to subsequent joins; the leave count, however, includes only the users leaving the system voluntarily while the cloud data are unchanged, and hence it is lower.

Additionally, in Fig. 7, the total communication sessions engaged in by the data owner and manager entities are shown, for another similar simulation run with the same parameters, to illustrate key differences. The costs for the cloud provider and mobile users (readers) were calculated, but did not vary across all the protocols and, hence, are not shown for conciseness. In the figures shown, the costs for the two cases of the proposed protocol (with and without a group key) overlap.

Furthermore, the sensitivity of the simulated performance of the system to the various parameter values was assessed, as shown in Table 8. The results are as expected:

- If the user population increases, the owner remains involved only in the first encryption and key setup

TABLE 7
The Parameters Used for the Simulation

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 8
Sensitivity of Simulation Results to Simulation Parameter Values

Parameter adjustment applying to each round	Owner	Mgr.	Cloud	Users
User population is 10×	-	+927%	+900%	+980%
P(user leaving) is 10×	-	-	+475%	-39%
P(user access) is 10×	-	-	-	+915%
P(replacement) is 10×	+100%	+8%	-75%	-75%

The parameter values from Table 7 were adopted, and then one of the probability-based values was modified in each case, as indicated, for the duration of the simulation. The proposed algorithm with the use of a group key was selected. The resulting differences in total computation for each entity are shown. Owing to the random nature of the simulation, only differences in excess of 5 percent are reported.

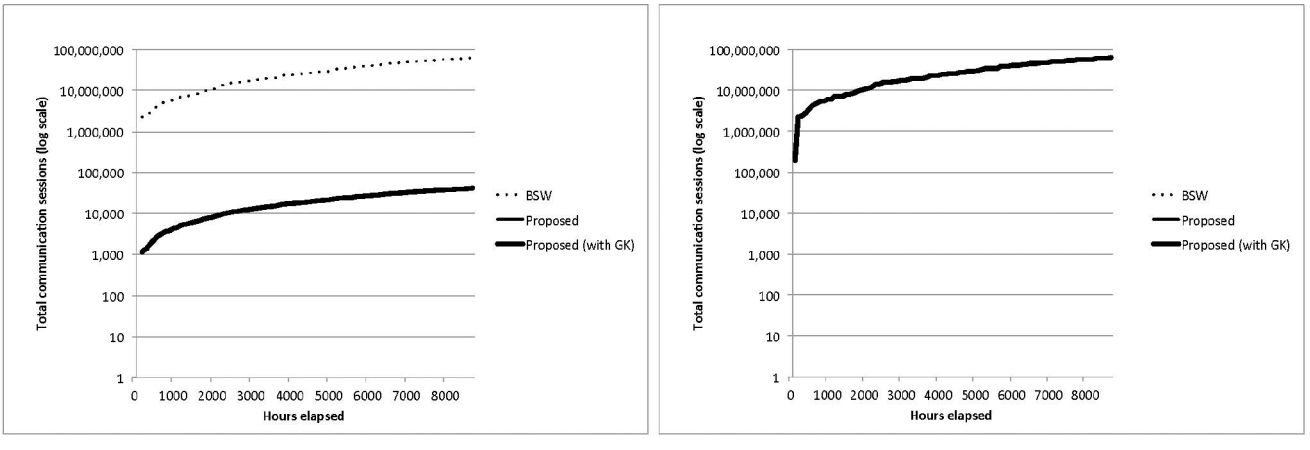
activity, and is not otherwise significantly affected in terms of its workload during normal operation of the system, including during the process of user revocation.

- If users become more likely to lose authorization immediately after consuming data, then the re-encryption process stemming from revocation incurs an additional (but proportionally smaller) load on the cloud provider, while the total load on all users becomes smaller because fewer decryptions of the same data are performed overall.
- If users access data more frequently, then there is a proportional increase in computational workload for users consisting of decryption work, which is unavoidable; the data owner is not involved in this activity, however.
- If cloud data have a shorter lifetime and requires more frequent replacement, then the data owner becomes significantly more engaged due to the associated encryption and key regeneration workload that is required. The system is most suitable, where the same data will be shared many times and not replaced on each consumption.

7.3 Discussion

The following observations may be made with respect to the results of the illustrated runs in Figs. 6 and 7, showing the various dominant roles in the system:

1. In the original BSW algorithm [4], the dominant computational and communication 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 Fig. 6a. The owner must also regenerate data secret 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, its scalability potential appears inadequate; intensive computation and data uploads serve to reduce the battery lifetime of the mobile device.
2. In the proposed algorithm *without* the use of a group key, the manager becomes responsible for the main computational workload of the key regeneration activity, which entails a pairing operation, as shown in Fig. 6b. It will also normally incur the communication cost of distributing data secret keys, removing the onus from the data owner, as shown in Figs. 7a and 7b. The manager is expected to be able to scale accordingly to meet the processing demands, but requires sufficient client infrastructure to do so (such as a private cloud), which may be uneconomical.



(a) The communication sessions for the mobile data owner.

(b) The communication sessions for the manager.

Fig. 7. The communication cost, for the data owner and manager, attained from the simulation.

TABLE 9
The *Incremental* Cost of Adding the Proposed Security Technique to a System for a Single Data Record, Assuming the Use of a Group Key, in Terms of the Additional Computation or the Number of Communication Sessions Required

Activity	Entity	Computation (ms)	Communication sessions (#)
Initial encryption, key setup	Owner	1.8	- (GSK, not DSK, sharing)
	Manager	1.0	1 (DSK sharing)
	Cloud	0.3	-
	Recipient	0	- (GSK, in addition to DSK, retrieval)
Decryption	Recipient	2.0	-
Re-encryption	Manager	0	1 (RK sharing)
	Cloud	0.4	-

The proposed protocol is compared to a system without data security. Timings are rounded to the nearest 10th of a second. The equipment assumed in use is the Galaxy Note II mobile phone for the owner and recipient, and an F1 instance running on a GAE cloud servlet for the cloud, as found in Table 6.

3. In the proposed algorithm *with* the use of a group key, the manager is still responsible for most of the key regeneration activity, but revocation is now handled through re-encryption via the group key, a task performed mainly by the cloud provider, as shown in Fig. 6c; this results in a much lower overall computational workload in the system. The additional decryption cost for the reader population is significant but acceptable, as shown in Fig. 6d. In practice, a decryption will only occur if data have been modified and have to be fetched from the cloud again. Hence, this algorithm is considered the best candidate for a highly scalable mobile cloud computing application within the system model described. Additionally, since the manager assumes the task of issuing a new version of the group secret key to all users (or they may even generate it themselves if the next version of the key can be derived from the previous version as suggested in [9]), the data owner is relieved from a significant communication burden due to its lack of participation in revocation activity; this will lead to a longer battery-powered lifetime. The collaboration of the managers and the cloud provider to relieve key management duties for the mobile owner and users is a key contribution of this work.

To gauge the incremental cost of adding the proposed security technique with the use of a group key to a system without security, the costs borne are estimated in Table 9 based on the benchmarks in Table 6. The data owner can take advantage of piggybacking by including its public keys and key components with the user data being uploaded to the cloud; in this way, the cloud provider does not incur additional communication sessions in the proposed protocol. The greatest penalty is on initial encryption for the mobile data owner, and it is still accomplished within 2 seconds. Importantly, the data owner experiences great savings in communication, as it does not need to send new versions of keys during key rotation and re-encryption in the proposed protocol, whenever revocation occurs; this activity is done by the manager. The owner is also not involved during regular decryption.

A table summarizing the proposed protocol's characteristics against other proxy re-encryption protocols found in related work, previously referred to in Section 2, appears in Table 10. As is evident, the proposed protocol offloads more activities from a mobile data owner to other network entities such as the manager, and minimizes the workload required for revocation by not being dependent on attributes; such an approach is especially suitable in a mobile cloud computing context.

8 Summary

A key management system has been proposed for secure data outsourcing applications, whereby attribute-based encryption effectively permits authorized users to access

TABLE 10
Comparison of Proposed Protocol (with the Use of a Group Key) to Related Work

Characteristic	Protocol in [14]	Protocol in [15]	Protocol in [16]	Protocol herein
System model:	Owner/authority, server	Owner, authority, CSP	Owner, CSP	Owner, manager, CSP
Cryptographic technique:	CP-ABE	CP-ABE	KP-ABE (requiring access structure for user)	Hybrid CP-ABE
Participating actors in user data encryption task:	Data owner	Data owner, attribute authority jointly (to offload access control)	Data owner	Data owner, manager, CSP jointly (to offload owner's computation)
Pairing operation required by owner on encryption:	✓	✓	✓	Offloaded to manager
Participating actors in re-encryption keygen task:	Data owner	Attribute authority	Data owner	Data owner or manager, by delegation
Mechanism for user revocation:	Multiple attribute keys regenerated	Multiple attribute keys regenerated	Multiple attribute keys regenerated	Single group key (GK) regenerated
Participating actors in cloud data re-encryption task:	CSP in lazy fashion	Attribute authority	CSP in lazy fashion	CSP in lazy fashion
Cloud-hosted metadata history for re-encryption task:	RKs and attributes updated by owner	None	RKs and attributes updated by owner	RKs and GK hashes updated by manager
Mechanism for decryption by recipient:	Process access subtree	Process dual access trees (attributes and user ID)	Process access subtree	Process access subtree and GK with manager's computational aid, requiring no extra pairing

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 cooperate 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. In particular, the user is not required to perform costly pairing operations; instead, they are delegated to the manager and cloud provider. Also, the manager computes the decryption key, not the data owner, and it assists with key distribution on behalf of the owner.

Furthermore, a hybrid protocol is proposed that optionally allows message encryption based on a group key, allowing the user membership to be further refined for highly sensitive data. Additionally, it allows re-encryption to occur, and thus revocation to become efficient without necessitating existing common remedies and their limitations; an example is the expiration of attributes specified in the attribute-based policy that leads to constant key updates as time elapses. The proposed protocol is 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 in a system where data updates and encryption activities are frequent and dominant. 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 Dr. Tysowski.

REFERENCES

[1] P.K. Tysowski and M.A. Hasan, "Hybrid Attribute-Based Encryption and Re-Encryption for Scalable Mobile Applications in Clouds," Technical Report 13, Centre for Applied Cryptographic Research (CACR), Univ. of Waterloo, 2013.

[2] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM,* vol. 26, no. 1, pp. 96-99, Jan. 1983.

[3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," *Proc. Ninth ACM SIGCOMM Conf. Internet Measurement Conf. (IMC '09),* pp. 280-293, 2009.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proc. IEEE Symp. Security and Privacy (SP '07),* pp. 321-334, 2007.

[5] A. Tassanaviboon and G. Gong, "OAuth and ABE Based Authorization in Semi-Trusted Cloud Computing: Aauth," *Proc. Second Int'l Workshop Data Intensive Computing in the Clouds (DataCloud-SC '11),* pp. 41-50, 2011.

[6] X. Liang, R. Lu, and X. Lin, "Ciphertext Policy Attribute Based Encryption with Efficient Revocation," Technical Report BBCR, Univ. of Waterloo, 2011.

[7] J. Hur and D.K. Noh, "Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 22, no. 7, pp. 1214-1221, July 2011.

[8] G. Zhao, C. Rong, J. Li, F. Zhang, and Y. Tang, "Trusted Data Sharing over Untrusted Cloud Storage Providers," *Proc. IEEE Second Int'l Conf. Cloud Computing Technology and Science (CLOUDCOM '10),* pp. 97-103, 2010.

[9] P.K. Tysowski and M.A. Hasan, "Towards Secure Communication for Highly Scalable Mobile Applications in Cloud Computing Systems," Technical Report 33, Centre for Applied Cryptographic Research (CACR), Univ. of Waterloo, 2011.

[10] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," *ACM Trans. Information and System Security,* vol. 9, pp. 1-30, Feb. 2006.

[11] S. Jahid, P. Mittal, and N. Borisov, "EASiER: Encryption-Based Access Control in Social Networks with Efficient Revocation," *Proc. Sixth ACM Symp. Information, Computer and Comm. Security (ASIACCS '11),* pp. 411-415, 2011.

[12] Q. Liu, G. Wang, and J. Wu, "Clock-Based Proxy Re-Encryption Scheme in Unreliable Clouds," *Proc. 41st Int'l Conf. Parallel Processing Workshops (ICPPW),* pp. 304-305, Sept. 2012.

[13] J.-M. Do, Y.-J. Song, and N. Park, "Attribute Based Proxy Re-Encryption for Data Confidentiality in Cloud Computing Environments," *Proc. First ACIS/JNU Int'l Conf. Computers, Networks, Systems and Industrial Eng. (CNSI),* pp. 248-251, May 2011.

[14] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute Based Data Sharing with Attribute Revocation," *Proc. Fifth ACM Symp. Information, Computer and Comm. Security (ASIACCS '10),* pp. 261-270, 2010.

[15] Y. Ming, L. Fan, H. Jing-Li, and W. Zhao-Li, "An Efficient Attribute Based Encryption Scheme with Revocation for Outsourced Data Sharing Control," *Proc. First Int'l Conf. Instrumentation, Measurement, Computer, Comm. and Control,* pp. 516-520, 2011.

[16] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," *Proc. IEEE INFOCOM '10,* pp. 534-542, 2010.

[17] K. Yang and X. Jia, "Attributed-Based Access Control for Multi-Authority Systems in Cloud Storage," *Proc. IEEE 32nd Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 536-545, 2012.

[18] K. Yang, X. Jia, K. Ren, and B. Zhang, "DAC-MACS: Effective Data Access Control for Multi-Authority Cloud Storage Systems," *Proc. IEEE INFOCOM,* pp. 2895-2903, 2013.

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

[20] A.D. Caro and V. Iovino, "jPBC: Java Pairing Based Cryptography," *Proc. IEEE Symp. Computers and Comm. (ISCC),* 2011.

[21] A.D. Caro, "Java Pairing-Based Cryptography Library," http://libeccio.dia.unisa.it/projects/jpbc/, 2012.

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

[23] G. Wang, Q. Liu, and J. Wu, "Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Storage Services," *Proc. 17th ACM Conf. Computer and Comm. Security (CCS '10),* pp. 735-737, 2010.

Piotr K. Tysowski received the BASc degree in computer engineering and the MASc and PhD degrees in electrical and computer engineering, all from the University of Waterloo in Canada, in 1998, 2001, and 2013, respectively. He also received the MBA degree from Wilfrid Laurier University in Canada in 2006. He received the NSERC Canada Graduate Scholarship (Doctoral) and University of Waterloo President's Graduate Scholarship. He was a lead software engineer and later a product manager in embedded mobile device software at Research In Motion (now BlackBerry) in Waterloo from 2001 to 2009. He has also previously worked in engineering at several other leading high-tech firms in Canada. He is an inventor of more than 30 US patents issued in the field of wireless communications. His research interests include mobile communications, cloud computing security and privacy, and software engineering. He is a licensed professional engineer of Ontario.

M. Anwarul Hasan received the BSc degree in electrical and electronic engineering, the MSc degree in computer engineering, both from the Bangladesh University of Engineering and Technology, in 1986 and 1988, respectively, and the PhD degree in electrical engineering from the University of Victoria in 1992. He joined the Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada in 1993 and has been a full professor since 2002. At the University of Waterloo, he is also a member of the Centre for Applied Cryptographic Research, the Centre for Wireless Communications, and the VLSI Research Group. He received the Raihan Memorial Gold Medal. At the University of Victoria, he received the President's Research Scholarship four times. At the University of Waterloo, he received a Faculty of Engineering Distinguished Performance Award in 2000, and Outstanding Performance Awards in 2004 and 2010. He served on the program and executive committees of several conferences. From 2000 to 2004, he was an associate editor of the *IEEE Transactions of Computers*. He is a licensed professional engineer of Ontario. He is a senior member of the IEEE.