Secure Data Retrieval Based on Ciphertext Policy Attribute-Based Encryption (CP-ABE) System for the DTNs

S. Roy, M. Chuah

Lehigh University
Bethlehem, PA, USA - 18015

Abstract. Mobile Nodes in some challenging network scenarios suffer from intermittent connectivity and frequent partitions e.g. battlefield and disaster recovery scenarios. Disruption Tolerant Network (DTN) technologies are designed to enable nodes in such environments to communicate with one another. Several application scenarios require a security design that provides fine grain access control to contents stored in storage nodes within a DTN or to contents of the messages routed through the network. In this paper, we propose an access control scheme which is based on the Ciphertext Policy Attributed-Based Encryption (CP-ABE) approach. Our scheme provides a flexible fine-grained access control such that the encrypted contents can only be accessed by authorized users. Two unique features our scheme provide are: (i) the incorporation of dynamic attributes whose value may change over time, and (ii) the revocation feature. We also provide some performance results from our implementation.

1 Introduction

The design of the current Internet service models is based on a few assumptions such as (a) the existence of an end-to-end path between a source and destination pair, and (b) low round-trip latency between any node pair. However, these assumptions do not hold in some emerging networks. Some examples [4] are: (i) battlefield ad-hoc networks in which wireless devices carried by soldiers operate in hostile environments where jamming, environmental factors and mobility may cause temporary disconnections, and (ii) vehicular ad-hoc networks where buses are equipped with wireless modems and have intermittent RF connectivity with one another.

In the above scenarios, an end-to-end path between a source and a destination pair may not always exist where the links between intermediate nodes may be opportunistic, predictably connectable, or periodically connected. To allow nodes to communicate with each other in these extreme networking environments, recently the research community has proposed a new architecture called the disruption tolerant network (DTN). Several DTN routing schemes [3, 5, 10, 14] have been proposed. Typically, the source node's message may need to wait in the intermediate nodes for substantial amount of time when there is no connection to the final destination. After the connection is eventually established, the message is delivered to the destination node.

In some application scenarios, there are some 'storage nodes' (which may be mobile or static) in the network where useful data is stored or replicated [6] so that other regular mobile nodes (also called users) can access the necessary information quickly and efficiently. A requirement in some security-critical applications is to design an access control system to protect the confidential data stored in the storage nodes or contents of the confidential messages routed through the network. As an example, in a battlefield DTN, a storage node may have some confidential information which should be accessed only by a member of 'Battalion 6' or a participant in 'Mission 3'. Several current solutions [7, 9] follow the traditional cryptographic-based approach where the contents are encrypted before being stored in storage nodes, and the decryption keys are distributed only to authorized users. In such approaches, flexibility and granularity of content access control relies heavily on the underlying cryptographic primitives being used. It is hard to balance between the complexity of key management and the granularity of access control using any solutions that are based on the conventional pairwise key or group key primitives. Thus, we still need to design a scalable solution that can provide fine-grain access control.

In this paper, we describe a CP-ABE based encryption scheme that provides fine-grained access control. In a CP-ABE scheme, each user is associated with a set of attributes based on which the user's private key is generated. Contents are encrypted under an access policy such that only those users whose attributes match the access policy are able to decrypt. Our scheme can provide not only fine-grained access control to each content object but also more sophisticated access control semantics e.g. "Captain or ((Battalion 6) or ((Mission 3) AND (NOT User 1))". Our solution builds on Bethencourt et al.'s [1] CP-ABE scheme. One of the major improvements achieved by our scheme over Bethencourt et al.'s [1] work is that our scheme can efficiently revoke one or multiple users. To introduce the revocation feature in our scheme, we modify Bethencourt et al.'s CP-ABE construction to incorporate the non-monotonic access structure proposed by Ostrovsky et al. [13]. We note that Ostrovsky et al. only used the non-monotonic access structure to design a Key Policy Attribute-Based Encryption (KP-ABE) system but not for a CP-ABE system. It is well known that CP-ABE systems are far better than KP-ABE systems in

terms of the power of the encryptor to control who has access to the data she encrypts [1].

In addition, our CP-ABE construction also addresses the scenario where both static and dynamic attributes may play a role in the data access policy. As an example, in a battlefield DTN, the soldiers can move from one strategic region to another region, i.e., 'location' of a node is a dynamic attribute. Some secret information may need to be available only to the soldiers who belong to 'Battalion A' (a static attribute), and are currently located in 'Region B' (a dynamic attribute).

Below we summarize the contributions made by this paper:

- We propose a CP-ABE scheme which supports non-monotonic access policy, so our scheme provides the revocation feature. We also implement the basic primitives of our scheme and report some performance results. We further improve the performance of our scheme by combining it with symmetric key systems. Moreover, we provide some ideas for hiding the access policy.
- We investigate the applicability of our CP-ABE scheme in security-critical DTNs. We further extend our scheme to incorporate static and dynamic attributes in the access policy.

Organization. The rest of this paper is organized as follows: Section 2 presents the preliminaries of the ABE systems. In Section 3, we provide our CP-ABE construction which supports the non-monotonic access structure. Section 4 explains how our CP-ABE system can be applied to ensure data confidentiality in the DTN environment, and we discuss the revocation issues in Section 5. In Section 6, we extend our CP-ABE construction to support static and dynamic attributes. Finally, we conclude in Section 7.

2 Attribute-Based Encryption (ABE) Systems

Recently, the research community has proposed Attribute-based Encryption (ABE) systems where encryption and decryption are determined by the attributes of the data and the recipients. An ABE cryptosystem is designed to enable fine-grained access control of the encrypted data. It allows the encryptor to attach attributes or policies to a message being encrypted so that only the receiver(s) who is (are) assigned compatible policies or attributes can decrypt it. Formally, the attributes can be considered as boolean variables with arbitrary labels, and the policies are expressed as conjunctions and disjunctions of attribute variables. The ABE systems can be viewed as a generalization of Identity Based Encryption (IBE) systems [2]. In IBE systems, only one attribute is used which is the identity of the receiver, whereas ABE systems enable the use of multiple attributes simultaneously.

In fact, current ABE schemes are built by cleverly combining the basic techniques of IBE with a linear secret sharing scheme. In these schemes, we can write the access policies in the form of a monotonic boolean formula over the attribute variables [8, 1]. We have two alternatives in enforcing the access policy. The access policy can be embedded in the private key of a user, which results in a cryptosystem called Key Policy ABE (KP-ABE) [8]. Alternatively, the access policy can be embedded in the ciphertext, which results in the Ciphertext Policy ABE (CP-ABE) system [1]. Both KP-ABE and CP-ABE systems ensure that a group of users cannot access any unauthorized data by colluding with each other.

Next, we summarize how the CP-ABE system works.

2.1 Ciphertext-policy ABE (CP-ABE)

In the CP-ABE scheme described in [1], each user is associated with a set of attributes and her private key is generated based on these attributes. When encrypting a message M, the encryptor specifies an access structure which is expressed in terms of a set of selected attributes for M. The message is then encrypted based on the access structure such that only those whose attributes satisfy this access structure can decrypt the message. Unauthorized users are not able to decrypt the ciphertext even if they collude. In [1], the access structure is sent in plaintext. A CP-ABE scheme consists of the following four algorithms:

1. Setup: This is a randomized algorithm that takes a security parameter as input, and outputs the public parameters PK and a master key MK. PK is used for encryption and MK is used to generate user secret keys and is known only to the central authority.
2. Encryption: This is a randomized algorithm that takes as input a message M, an access structure T, and the public parameters PK. It outputs the ciphertext CT.
3. KenGen: This is a randomized algorithm that takes as input the set of a user (say X)'s attributes S_X, the master key MK and outputs a secret key SK that identifies with S_X.
4. Decryption: This algorithm takes as input the ciphertext CT, a secret key SK for an attribute set S_X. If S_X satisfies the access structure embedded in CT, it will return the original message M.

The security authority holds the master key MK and publishes the public parameters PK. Using PK, a user can encrypt an arbitrary message, M using any abitrary access structure T based on a set of attributes, S. It is worth noting that there is no specific public key for a user in this system contrary to common public-private key system such as RSA. Moreover, the size of the ciphertext is of $O(S_T)$, if there are S_T nodes in the access tree, T. A user's secret key is determined by the subset of attributes

she owns. Hence, her private key size is $O(\tau)$ if she owns only τ attributes.

The work in Bethencourt et al.'s [1] CP-ABE system only supports monotonic access structure in the access policy. In other words, only positive attributes can be present in the access tree, which limits the expressiveness of the access policy. We would like to have a feature that allows us to specify negative attributes in the access policy, that refers to the absence of the positive attributes. As an example, a short access structure such as 'NOT Mission 2' is satisfied by a user X's set of attributes, S_X only if the attribute 'Mission 2' is not present in S_X.

3 A CP-ABE system with the non-monotonic access structure

We construct a CP-ABE scheme which supports the use of positive and negative attributes. Our construction is built on Bethencourt et al.'s work [1] and Ostrovsky et al.'s [13] work. Specifically, we modify Bethencourt et al.'s CP-ABE construction to accommodate a non-monotonic access structure. We note that the non-monotonic access structure proposed in Ostrovsky et al.'s scheme is only for KP-ABE systems.

Recall that in a CP-ABE cryptosystem, a plaintext M is encrypted based on an access tree structure T to generate a ciphertext CT. The private key of user X is identified with the set of X's attributes, S_X. A private key, SK will be able to decrypt CT if the set of attributes, S_X assigned to SK satisfies the access tree T.

In our construction, positive attributes (e.g. 'Commander') as well as negative attributes (e.g. 'NOT Mission 2') can be present in T. However, private key components are only assigned to positive attributes. That is, for any X, S_X does not contain any negative attribute.

3.1 Our Construction

To describe our construction, we use the terminology and language as in [1, 13]. We start with describing the access tree, T.

Each leaf node of the access tree T represents either a positive or a negative attribute. Each internal node of T represents a threshold gate, which can be an 'AND' gate or an 'OR' gate in special cases. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. We note that a threshold gate represents an OR gate if $k_x = 1$, and an AND gate if $k_x = num_x$. We denote the parent of node x by $parent(x)$. The function $att(x)$, which is defined only for the leaf nodes, denotes the attribute associated with a leaf node x.

In the access tree, we also define an ordering among the child nodes of every node. The children of node x are numbered from 1 to num_x. We denote such a number associated with node x by $index(x)$.

We map each attribute to a unique integer in Z_p^*, which could be done by using a collision-resistant hash function $F : \{0,1\}^* \rightarrow Z_p^*$. As an example, if the attribute in a leaf node x is 'Commander', then $att(x) = F(\text{'Commander'}) \in Z_p^*$, and if the attribute in a leaf node x is 'NOT Mission 2', then $att(x) = F(\text{'Mission 2'}) \in Z_p^*$. Note that the 'NOT' part of an attribute in a node x is not used to compute $att(x)$. In the rest of Section 3, we interchangeably use 'attribute A' and the corresponding integer $i = F(A)$.

We denote by T_x the subtree of T rooted at node x. If a set of attributes S (represented by a subset of Z_p^*) satisfies the access tree T_x , we denote it as $T_x(S) = 1$. We compute $T_x(S)$ recursively as follows. If x is a non-leaf node, evaluate $T_{c_x}(S)$ for each child node c_x of x. $T_x(S)$ returns 1 if and only if at least k_x child nodes return 1, where k_x is the threshold value of node x. If x is a leaf node and denotes a positive attribute, then $T_x(S)$ returns 1 if and only if $att(x) \in S$. On the other hand, if x is a leaf node and denotes a negative attribute, then $T_x(S)$ returns 1 if and only if $att(x) \notin S$.

Let G_0 be a bilinear group of prime order p, and let g be a generator of G_0. In addition, let $e : G_0 \times G_0 \rightarrow G_1$ denote the bilinear map. We also use a function $H : Z_p^* \rightarrow G_0$ to map any attribute to an element in G_0 , where $H(i) = g^i$. Our crytosystem consists of the following alrogithms.

Setup. This algorithm chooses three random exponents $a, b, \gamma \in Z_p$. In the basic construction, a parameter d specifies how many attributes every private key has. (This constraint will be removed later.) One polynomial $v(x)$ of degree d is chosen at random subject to the constraint that $v(0) = b$. The public parameters are as follows:

$$PK = [G_0, g, h = g^b, h^\gamma, f = g^{1/\gamma}, e(h,g)^a,$$

$$\{g^{v(0)}, g^{v(1)}, \ldots, g^{v(d)}\}].$$

We note that h is the same as $g^{v(0)}$, but for the sake of clarity, we include both of them in the description of PK. The master key MK is $[b, \gamma, g^a]$.

Encryption (PK, M, T). The encryption algorithm encrypts a message M, $M \in G_1$ under the tree access structure T. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree T. These polynomials are chosen in the following way in a top-down manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Starting with the root node R the algorithm chooses a random $s \in Z_p$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x.

Let, $Y1$ and $Y2$ be the set of the leaf nodes in T with positive attributes and negative attributes, respectively. For each node $y \in Y2$, we randomly choose u_y from Z_p. We define a function $V(i) : Z_p \rightarrow G_0$ as $V(i) = g^{v(i)}$.

Note that though $v(i)$ is not known to the encryptor, she can compute $V(i)$ by interpolation using the set $\{g^{v(0)}, g^{v(1)}, \ldots, g^{v(d)}\}$ available in the public key, PK. We also recall that function F maps each attribute to an element in Z_p^*. The ciphertext CT is then constructed by providing the tree access structure T and computing the other components as follows.

$$CT = [T, C1 = Me(h,g)^{as}, C2 = (h^\gamma)^s,$$
$$\forall \text{ nodes } y \in Y1 \ : \ C1_y = g^{q_y(0)}, C2_y = H(i)^{q_y(0)}$$
$$\text{where } i = F(att(y)), \text{ and}$$
$$\forall \text{ nodes } y \in Y2 \ : \ C3_y = h^{q_y(0)+u_y}, C4_y = (V(i))^{u_y},$$
$$C5_y = g^{u_y} \text{ where } i = F(att(y))].$$

We note that the public key components $e(h,g)^a$ is used to compute $C1$, and h^γ is used to compute $C2$.

KeyGen (MK, S). The key generation algorithm will take as input a set of attributes $S \subset Z_p^*$ and output a private key that identifies with S. The algorithm first chooses a random $r \in Z_p$, and then random $r_j \in Z_p$ for each attribute $j \in S$. Then it computes the key as

$$SK =$$
$$[D = g^{(a+r)/\gamma}, D1 = g^r, \forall j \in S \ :$$
$$D1_j = h^r.H(j)^{r_j}, D2_j = g^{r_j}, D3_j = (V(j))^r]$$

We assume that the the size of set S is fixed, which is d, the maximum number of attributes associated with a user. In practice, we can get around this restriction as follows. If an user with Id X is associated with d' attributes where $d' < d$, then we can just add $(d - d')$ filler attributes in X's set of attributes such as "$X : Filler1$", "$X : Filler2$", and others. We assume that d is a small constant. To reduce the overhead when d is large, we can adapt the optimization techniques present in Ostrovsky et al.'s work [13].

Delegate$(SK, \bar{S})$. The delegation algorithm takes in a secret key SK, which is for a set S of attributes, and another set $\bar{S}$ such that $\bar{S} \subset S$. The algorithm chooses random $\bar{r}$ and $\bar{r_k}$ $\forall k \in \bar{S}$. Then it creates a new secret key as

$$\bar{SK} = [\bar{D} = D.f^{\bar{r}}, \bar{D1} = g^r.g^{\bar{r}}, \forall k \in \bar{S} \ :$$
$$\bar{D1}_k = D1_k.h^{\bar{r}}.H(k)^{\bar{r_k}}, \bar{D2}_k = D2_k.g^{\bar{r_k}},$$
$$\bar{D3}_k = D3_k.(V(k))^{\bar{r}}].$$

Decrypt(CT, SK). We specify our decryption procedure as a recursive algorithm. For ease of exposition we present the simplest form of the decryption algorithm. We may adapt the optimizations reported in [1] to design a more efficient decryption algorithm; however, we do not discuss that here.

We first define a recursive algorithm DecryptNode(CT, SK, x) that takes as input a ciphertext CT, a private key SK, which is associated with a set S of attributes, and a node x from T .

If the node x is a leaf node, then we let $i = att(x)$. Note that x can contain a positive or negative attribute. If x corresponds to a positive attribute and $i \in S$, then

$$DecryptNode(CT, SK, x) = \frac{e(D1_i, C1_x)}{e(D2_i, C2_x)}$$

$$= \frac{e(h^r.H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} = \frac{e(h^r, g^{q_x(0)}) \times e(H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})}$$
$$= e(h^r, g^{q_x(0)}) = e(h,g)^{rq_x(0)}.$$

If x corresponds to a positive attribute and $i \notin S$, then we define $DecryptNode(CT, SK, x) = \perp$, which indicates that node x is not satisfied by SK.

On the other hand, if node x corresponds to a negative attribute and $i = att(x) \notin S$, then we say that node x is satisfied by S. As an example, if the attribute in node x is 'NOT Mission 2', then node x is satisfied by S only if S does not contain $i = F$('Mission 2').

The decryption procedure first computes the following: We consider the set $S' = S \cup i$. Note that $|S'| = d+1$ and recall that the degree of polynomial v underlying function V is d. Using the points in S' as an interpolation set, we compute Lagrangian coefficients $\{\sigma_j\}_{j \in S'}$ such that $\sum_{j \in S'} \sigma_j v(j) = v(0) = b$. Now the following function is computed.

$$DecryptNode(CT, SK, x)$$
$$= \frac{e(D1, C3_x)}{e(C5_x, \prod_{j \in S} (D3_j)^{\sigma_j}).e(D1, C4_x)^{\sigma_i}}$$
$$= \frac{e(g^r, h^{q_x(0)+u_i})}{e(g^{u_i}, \prod_{j \in S} (V(j))^{r\sigma_j}).e(g^r, V(i)^{u_i})^{\sigma_i}}$$
$$= \frac{e(g^r, h^{q_x(0)}).e(g^r, h^{u_i})}{e(g^{u_i}, g^{\sum_{j \in S} r\sigma_j v(j)}).e(g^r, g^{u_i v(i) \sigma_i})}$$
$$= \frac{e(h,g)^{rq_x(0)}.e(g,g)^{rbu_i}}{e(g,g)^{u_i r \sum_{j \in S'} \sigma_j v(j)}}$$
$$= e(h,g)^{rq_x(0)}$$

If x corresponds to a negative attribute and $i \in S$, then we define $DecryptNode(CT, SK, x) = \perp$.

We now recursively compute $DecryptNode(CT, SK, x)$ when x is a non-leaf node. For all nodes c that are children of x, we compute $DecryptNode(CT, SK, c)$ and store the output as L_c. Let U_x be an arbitrary k_x-sized set of child nodes c such that $L_c \neq \perp$. If no such set exists then the node x was not satisfied and the $DecryptNode(CT, SK, x)$ is set to $\perp$. Otherwise, using Langrange interpolation formula Δ we compute L_x where $\Delta_{m,P}(z) = \prod_{j \in P, \ j \neq m} \frac{z-j}{m-j}$ as follows :

$$L_x = \prod_{c \in U_x} L_c^{\Delta_{m,U'_x}(0)}, \ \text{where } m = index(c)$$
$$\text{and } U'_x = \{index(c) : c \in U_x\}$$

$$= e(h,g)^{rq_x(0)} \quad .$$

Now that we have defined our function $DecryptNode$, we can define the decryption algorithm. The algorithm begins by simply calling the function on the root node R of the access tree T. If the tree is satisfied by S, we set $A = DecryptNode(CT, SK, R) = e(h,g)^{rq_R(0)} = e(h,g)^{rs}$. The algorithm now decrypts the ciphertext by computing

$$C1/(e(C2,D)/A) = C1/(e(h^{\gamma s}, g^{(a+r)/\gamma})/e(h,g)^{rs})$$

$$= \frac{M.e(h,g)^{as}}{e(h,g)^{as}} = M$$

Finally, we note that the above CP-ABE scheme also supports any access tree with 'NOT' gate as an internal node. We observe that repeated application of DeMorgan's law can push down the 'NOT's to leaf nodes transforming the original tree to another one where internal gates are only threshold gates.

3.2 Efficiency of Our Construction

We now discuss the complexity of the algorithms presented above, such as setup, encryption and others. We note that the most expensive operations are to compute a pairing and the exponentiations. The Setup algorithm will require one pairing operation and $(d+3)$ exponentiations, where d is the number of attributes present in each private key. Each public key contains $(d+5)$ group elements. The Encryption algorithm will require no pairing operation. However, it will require two exponentiations to generate $C1$ and $C2$. Moreover, for each positive leaf node in the access tree, we will require two exponentiations. For each negative leaf node, the Encryption algorithm will require $(d+4)$ exponentiations among which $(d+1)$ exponentiations are needed for computing $V(i)$ by Langrange interpolation. If μ is the number of positive leaf nodes and ν is the number of negative leaf nodes, the ciphertext contains $(2\mu+3\nu+2)$ group elements. The KeyGen algorithm will require two exponentiations to generate D and $D1$. Moreover, for each attribute, the KeyGen algorithm will perform four exponentiations, so the total cost of KeyGen is $(4d+2)$ exponentiations. Each private key contains $(3d+2)$ group elements. The complexity of the Decrypt algorithm depends on the specific private key and the access tree pair as it is possible that a large and complicated access tree could be satisfied with a small number of attributes in the private key via a shortcut way. For each positive leaf in the access tree that needs to be decrypted we require two pairing operations whereas for each negative leaf that needs to be decrypted we require $(d+3)$ exponentiations and three pairing operations. The Decrypt algorithm also requires one exponentiation for each internal node in the access tree along a path for such a leaf node to the root.

We have implemented our construction using the libraries previously developed by other researchers [11, 1]. We now discuss the performance of our implementation running on a Intel Core (TM) 2 Quad 2.4 GHz workstation. Like the previous researcher [1], our implementation uses a 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$ over a 512-bit finite field. We report the time taken by the Setup, Encryption, KeyGen, and Decrypt algorithms. The reported value is the average of 30 independent runs. We observe that in each case, the 95% confidence intervals are within 2% of the reported value.

With $d = 20$, the time taken by the Setup algorithm was 0.076 sec. With $d = 20$, the Encryption algorithm took 1.332 sec when the number of positive attributes in the access tree is $\mu = 15$ and number of negative attributes is $\nu = 10$. With $\mu = 7$ and $\nu = 3$, the Encryption algorithm took 0.672 sec. With $d = 20$, the time taken by the KeyGen algorithm was 1.542 sec. With $d = 20$, the Decrypt algorithm took 1.980 sec when the number of positive attributes matched with the private key is $\mu' = 15$ and number of negative attributes matched is $\nu' = 10$. With $\mu' = 7$ and $\nu' = 3$, the Decrypt algorithm took 0.844 sec.

3.3 Using our CP-ABE System together with Symmetric Key systems

We observe that the maximum size (λ) of the message M which can be encrypted by the above construction is the same as the size of the elliptic curve group used. As an example, in our implementation, $\lambda = 160$-bit. To encrypt any larger message we need to divide the message into blocks of length λ and then encrypt these blocks one by one. We observe that it is expensive to encrypt a big file directly using the CP-ABE scheme, which involves costly operations such as exponentiation and Weil pairing. To reduce the encryption/decryption cost, we propose to combine our CP-ABE system with symmetric key cryptosystems as described below.

For a data object π (which may be a file) of size larger than λ, the encryptor generates a symmetric key ρ and encrypts the object π with ρ to generate $E_\rho(\pi)$ using a symmetric encryption scheme, E. E can be a block cipher such as AES. Then the symmetric key ρ is encrypted using our CP-ABE encryption scheme presented above. Then, the encrypted object, $E_\rho(\pi)$, and the encrypted key is sent to the storage nodes. The authorized users can retrieve both pieces of information from the storage nodes, then extract the ρ using the CP-ABE decryption scheme and later use ρ to obtain the original object π.

We now discuss the execution time saving by such a scheme. The total time required if we do not combine our scheme with a symmetric key system is $\phi_1 = \frac{L}{\lambda}.t_1$, where t_1 is the time taken by one CP-ABE encryption and L is the size of the file to be encrypted. On the other hand, the total time required if we combine our scheme with a block cipher system is $\phi_2 = \frac{L}{\lambda'}.t_2 + t_1$, where t_2 is the time taken by one symmetric encryption and λ' is the

size of one block of the block cipher system. As $t_2 << t_1$, $\phi_2 << \phi_1$ for a large file.

As an example, in our implementation, we use AES with $\lambda' = 128$ bit block size as the block cipher. For one setting, we have measured that t_1 is 0.670 sec and t_2 is negligible in comparison to it when the file size varies from 1k, 10k to 100k bytes. Thus, we obtain the following : $\phi_1 = 4.188, 41.875, 418.750$ sec, and $\phi_2 = 0.670, 0.670, 0.670$ sec for the three file sizes respectively. ϕ_2 remains virtually unchanged as the file size grows to 100k. We only observed a slightly larger $\phi_2 = 0.740$ sec when we try a very large file of 10GB.

3.4 Keeping the Access Policy Hidden

We note that in our construction the ciphertext, CT contains the access tree, T which is in the clear text. Researchers always wish to make the access policy hidden, if possible, so that the adversary does not have any clue about the set of the intended receivers. We now make a little modification to our construction, which could keep most of the access policy used in the ciphertext hidden.

The modification of our construction is only in stating the access policy, which is described below while the rest of the construction remains the same. Our modified access tree, T does not contain any specific value of an attribute—instead, T specifies only the attribute name. However, the ciphertext fields (e.g., $C1_y$, $C4_y$) are computed using both the attribute names and the specific values. During decryption, the receiver X appends to each matched attribute name in T its own value. So, X is able to decrypt if and only if X is an authorized user . Assuming an attribute could have multiple possible values in the application domain, the modified scheme does not reveal the policy to the adversary yet enables the intended retrieval.

As an example, say in a battlefield DTN there are 10 battalions deployed and each battalion has 4 companies. The battlefield is divided into 100 regions which is also an attribute. So, the attribute 'Battalion' can have 10 values such as 'A', 'B', and others, the attribute 'Company' can possibly possess 4 values such as '1', '2', and others, and the attribute 'Region' can be assigned values such as '1', '2', etc. Let the access policy is (('Battalion A' AND 'Company 2') OR 'Region 1'). In our modified scheme, the access tree T which is in the clear text would read (('Battalion' AND 'Company') OR 'Region').

We note that there is a recent work by Yu et al. [15], which proposes a CP-ABE scheme with Hidden Policy. We compare our modified scheme with Yu et al.'s scheme. Yu et al.'s work assumes that each attribute is binary and ensures that the adversary cannot conclude which binary value is present in the access policy. The limitation of this work is it does not support a general access policy while it assumes that the access policy is the AND of all of the attributes. The public key, private key, ciphertext is of size $O(N)$, where N is the total number of attributes present in the application domain. All operations such as Setup, Encryption, KeyGen, and Decrypt require $O(N)$ computations. On the other hand, our scheme not only supports general access policy but hides most of the access policy. The public and private key is of size $O(d)$, where d is the number of attributes present in each private key, which is typically a small constant. Operations such as Setup, and KeyGen require $O(d)$ computations while the cost of Encryption and Decrypt operations is application specific.

4 Securing Data in DTNs

Now we discuss how confidential data can be protected from unauthorized users in a DTN environment using our CP-ABE scheme. First we provide our network and security model and then discuss how our scheme can be used to enforce a fine-grained access control.

4.1 The Network and Security Model

Without loss of generality, we assume that each user is a node in the DTN network. A user should be able to access only those data that are allowed within her role. As an example, in a battlefield DTN the 'captain' may have more access privilege than a common 'soldier' does, and a 'soldier' should not be allowed to access some data which is beyond her policy.

There may exist some 'storage nodes' (mobile or static) in the network where useful data is stored or replicated which other regular mobile nodes (also called users) might need to access. In another situation, there may be a user in the network who wants to send some confidential message to an intended subset of users. We observe that the above two cases are equivalent from the security perspective while we want to ensure that an unauthorized user cannot access the confidential information.

We assume the existence of a central authority (CA) in our setting, which can be realized by deploying a mobile key server (MKS).

4.2 Applying CP-ABE

The basic idea is that the sender encrypts the data according to the access tree, T. Each leaf node in T represents an attribute, e.g., the context, the degree of importance, the creation time, the owner, etc. The non-leaf nodes in T are threshold gates of which AND or OR is a special case. Each user X is assigned a set of attributes, S_X by the central authority. The ciphertext, CT is labeled with an access tree T, and only if a user's set of attributes, S_X satisfy T, she will be able to decrypt CT.

It is assumed that each user receives the PK from the central authority, CA during the bootstrap phase. During the setup time each user also gets her private key (decryption key) determined by her set of attributes from the

CA. As a result, the user does not need to contact CA to obtain a grant every time she tries to decrypt some ciphertext, i.e., the user's decryption power is embedded in her private key. For the time being we assume that each user only have static attributes which does not change over time. In Section 6, we discuss how to handle users with dynamic attributes while private keys have to be refreshed.

Let us consider a battlefield scenario where several battalions of army are participating in some missions. Each battalion has a 'captain' and several subordinate 'soldiers'. Each 'captain' or 'soldier' is potentially moving with a node capable of wireless communication. There may or may not be any additional communication infrastructure such as a gateway or a moving ferry. The members of the battalions are geographically intermingled and form a sparse network where each member is a node in the DTN layer. We refer to this scenario as Scenario 1.

Suppose the 'captain' of Battalion 6 wants to send a confidential command message M to an intended audience. To this end, the sender encrypts M with the access tree as follows: (('Battalion 6' AND 'Mission 3') OR 'Captain'). An example of a few users and their sets of attributes is given below.

- User 1 (the 'captain' of Battalion 4): 'Battalion 4', 'Captain'
- User 2 (a 'soldier' of Battalion 6): 'Battalion 6', 'Soldier', 'Mission 3'
- User 3 (a 'soldier' of Battalion 4): 'Battalion 4', 'Soldier', 'Mission 3'
- User 4 (a 'soldier' of Battalion 4): 'Battalion 4', 'Soldier', 'Mission 3'

We observe that the attributes of users 1, and 2 satisfy the access structure of the ciphertext and hence these users will be able to decrypt it. However, as the access structure is not satisfied by the set of attributes of user 3 or 4, they will not be able to decrypt this message.

If, in the above example, the access policy of the ciphertext is ('Battalion 6' AND 'Mission 3'), then only user 2 will be able to decrypt M but neither user 1 nor user 3 nor user 4 will.

We note that using this scheme, it is possible to embed an arbitrarily complex access policy in a ciphertext. The internal nodes of the access tree are threshold gates and the leaves are the attributes. In general, a threshold gate is of the nature of k-of-n, $1 \leq k \leq n$.

Moreover, the key delegation property of CP-ABE proves to be very useful in DTNs. This property allows any user that has a key for a set of attributes, S to derive a key for a smaller set of attributes, $S' \in S$. As an example, there may exist a regional key server, RKS_i in region i (possibly, one in each region) which can receive the credentials from the MKS and can act as its delegate in the corresponding region.

5 Revocation Issues

Revocation of users in cryptosystems is a well studied but nontrivial problem [12]. Revocation is even more challenging in attribute-based systems, given that each attribute possibly belongs to multiple different users, whereas in traditional PKI systems public/private key pairs are uniquely associated with a single user. In principle, in an ABE system, attributes, not users or keys, are revoked.

Now we discuss how the revocation feature can be incorporated. A simple but constrained solution is to include a time attribute. This solution would require each message to be encrypted with a modified access tree T', which is constructed by augmenting the original access tree T with an additional time attribute. The time attribute, ζ represents the current 'time period'. Formally, the new access structure T' is as follows: $T' = (T \text{ AND } \zeta)$. For example, ζ can be the 'date' attribute whose value changes once every day. It is assumed that each non-revoked user receives his fresh private keys corresponding to the 'date' attribute once everyday directly from the mobile key server MKS (which is the central authority) or via the regional delegates. With a hierarchical access structure, the key delegation property of CP-ABE can be exploited to reduce the dependency on the central authority for issuing the new private keys to all users every time interval.

There are significant trade-offs between the extra load incurred by the authority for generating and communicating the new keys to the users and the amount of time that can elapse before a revoked user can be effectively purged. This above solution has the following problems: (a) Each user X needs to periodically receive from the central authority the fresh private key corresponding to the time attribute, otherwise X will not be able to decrypt any message. (b) It is a lazy revocation technique—the revoked user is not purged from the system until the current time period expires. (c) This scheme requires an implicit time synchronization (a loose time synchronization may be sufficient) among the authority and the users. To address these issues, below we present a more practical revocation scheme.

5.1 Exploiting Non-Monotonic Access Structures

Our CP-ABE scheme, discussed in Section 3, can handle any access structure that can be represented by a Boolean formula involving AND, OR, NOT, and threshold operations. In particular, using a non-monotonic access structure it is possible to represent *negative* constraints in the access policy. Let us assume that each user has an attribute that represents his own ID. Then, the basic idea for revoking users using the non-monotonic access structure is to attach a negative constraint to the ciphertext's access policy which include the IDs of the revoked users.

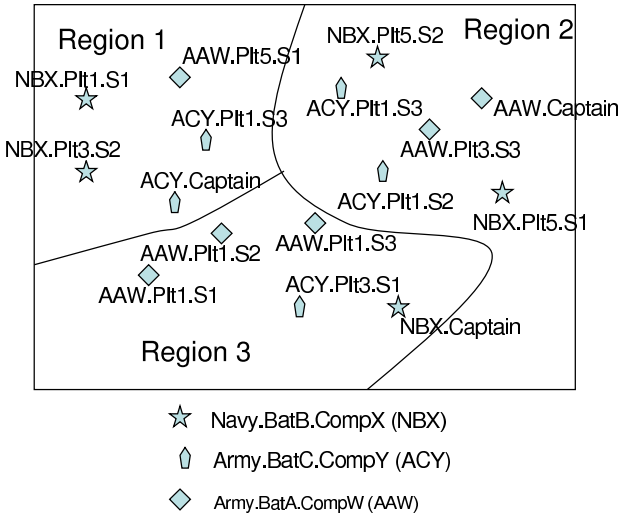

Fig. 1. A battlefield DTN where several members of different ranks from US Army and Navy are deployed. Each participant is labeled with her corresponding rank. As an example, if a soldier $S1$ is from Platoon 3 belonging to Company Y of Battalion C in the US Army, then her label is ACY.Plt3.S1. If a sender X wants to send a confidential message to a subset of users, then X embeds the corresponding access policy in the ciphertext. As an example, a message encrypted with the access policy such as ((‘ACY’ OR ‘AAW’) AND ‘Region 2’) can be decrypted by ACY.Plt1.S3, ACY.Plt1.S2, AAW.Captain, and AAW.Plt3.S3.

Let us consider an example with 4 users where each user’s set of attributes is given below:

- User 1 (the ‘captain’ of Battalion 4): ‘Battalion 4’, ‘Captain’, ‘User 1’
- User 2 (a ‘soldier’ of Battalion 6): ‘Battalion 6’, ‘Soldier’, ‘Mission 3’, ‘User 2’
- User 3 (a ‘soldier’ of Battalion 4): ‘Battalion 4’, ‘Soldier’, ‘Mission 3’, ‘User 3’
- User 4 (a ‘soldier’ of Battalion 4): ‘Battalion 4’, ‘Soldier’, ‘Mission 3’, ‘User 4’

Now if the ‘captain’ of Battalion 6 does not want to allow User 2 to read his command message M, he encrypts M with the following access policy: ((((‘Battalion 6’ AND ‘Mission 3’) OR ‘Captain’) AND (NOT ‘User 2’)). Thus, User 2 is effectively revoked from the system.

In general, this revocation technique would require each message to be encrypted with a modified access tree T', which is constructed by augmenting the original access tree T with an additional list of revoked user IDs. Formally, the new access structure T' is as follows:

$$T' = (T \text{ AND } ((\text{NOT User } X_1) \text{ AND } (\text{NOT User } X_2)$$

$$\ldots \text{ AND } (\text{NOT User } X_m))),$$

where users $X_1, X_2, \ldots, X_m$ have been revoked.

We observe that if m users have to be revoked, the ciphertext’s access tree will contain additional m leaf nodes.

This scheme also allows us to revoke a set of attributes (which may implicitly represent a set of users) instead of an explicit list of users. In the above example, augmenting the original access formula T with (NOT (‘Captain’ AND ‘Battalion 4’)) will effectively revoke User 1 from the system. The modified access structure will be

$$T' = ((((\text{‘Battalion 6’ AND ‘Mission3’})\text{OR ‘Captain’})$$

$$\text{AND (NOT (‘Captain’ AND ‘Battalion 4’))}.$$

6 Handling the Static and Dynamic Attributes in DTNs

We assume that the attributes of the DTN users (which are also called nodes) in our CP-ABE system are of two kinds—(i) static attributes whose value remain unchanged for long time, and (ii) dynamic attributes which need to be updated periodically. Figure 1 illustrates a battlefield DTN sceneraio where a hierarchy of the static attributes are used to specify the participating nodes. In addition, the whole network is divided into several regions. The nodes move around and hence the nodes’ locations (which is a dynamic attribute) change over time. A node can be in ‘Region k1’ at time t_1 and in ‘Region k2’ at time t_2. For example, the set of static attributes of Bob, which is {‘Soldier’, ‘Platoon B’, ‘Battalion A’, ‘Army’ and ‘US-Defense’} does not change over time whereas his dynamic attribute ‘location’ changes from Region $k1$ at time t_1 to Region $k3$ at time t_2.

Now, the problem we need to address is as follows: How to enable an authorized user X to decrypt a message which has been encrypted using an access tree consisting of some static as well as dynamic attributes?

We observe that X needs to be periodically given the private key corresponding to the current value of each of

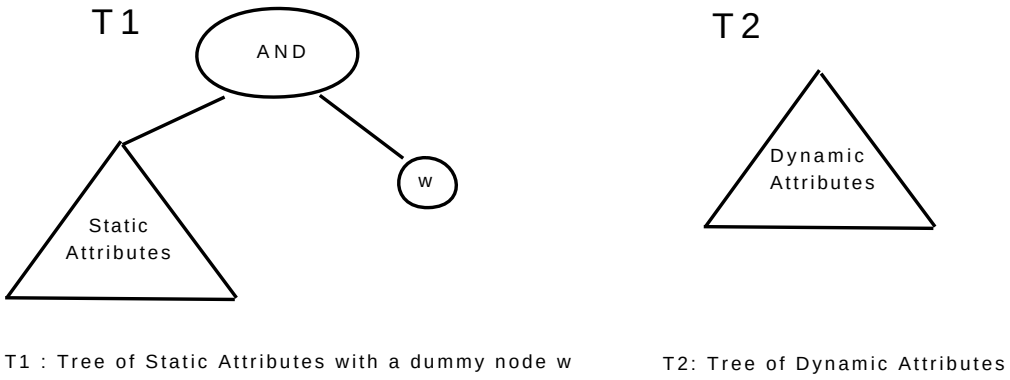

Fig. 2. Designing two separate access trees—one represents the static attributes and another represents the dynamic attributes.

its dynamic attributes. We assume that once a user qualifies for a new value of a dynamic attribute, e.g. it has entered into a new 'Region', the user can get the credentials for the new attribute from the local authority, RKS_k. As in a CP-ABE scheme the private keys are always randomized before they are delivered, it is not possible for the user X to combine the newly obtained key with the static part of the previous key. This is because a secret-sharing technique is used in the design of any CP-ABE scheme to prevent collusion attacks.

One straightforward solution to the above problem is as follows: Once the dynamic attribute of a user changes, the user is given a new private key for both of the static and dynamic attributes. This solution is expensive and it has a critical problem—the user has to communicate with the MKS (Mobile Key Server which is the central authority) each time any of its dynamic attributes is changed. We do not further discuss this inefficient solution.

Now, we propose another two solutions to address the above problem, which are applicable in different scenarios. In the first solution, we assume that the security policy is such that the local authority, RKS_k should have the privilege to decrypt all of the messages which have been delivered to the local nodes in region k. As an example, say the access policy embedded in a ciphertext is "('Captain' OR 'Mission 3' AND 'Region k')". If the set of the local authority RKS_k's attributes contain all of these 3 attributes, RKS_k can delegate the appropriate credential to each local user X once X comes under the jurisdiction of RKS_k. That is, if a captain X can prove her identity (or a user Y can prove her membership of 'Mission 3') to RKS_k, it can delegate to X (or Y) a private key which contains 'Captain', and 'Region k' (or 'Mission 3' and 'Region k'). However, this assumption is not too practical in many real-life scenarios, and hence in the rest of this section we focus on our second solution which is applicable in a more general setting—the set of a user's attributes does not necessarily belong to the set of the local authority's attributes.

Recall that in CP-ABE system, an access structure is used to encrypt any message. We assume that the access policy can be written in such a way that the set of static and dynamic attributes are connected by an 'AND' gate, which, in our belief, is the common case in practical scenarios. As an example, "(('Captain' OR 'Mission 3') AND 'Region k')" is such an access policy. We propose to divide the access structure in two distinct parts—one tree, $T1$ with the static attributes and another tree, $T2$ with the dynamic attributes as shown in Figure 2. In $T1$, we introduce a dummy leaf node, named w, which represents a dummy attribute on behalf of the whole set of dynamic attributes present in $T2$. Node w is considered to be a positive attribute and it is connected to the root of the other part of $T1$ by an 'AND' gate. Let the message to be encrypted be M. The encryptor X executes the function $Encryption(M, PK, T1)$ to generate the ciphertext CT. From the construction of our *Encryption* algorithm presented in Section 3, we see that there is a part in CT which corresponds to the leaf node w in $T1$. Let us call that part CT_w which is $[C1_w, C2_w]$. The rest of CT is referred to by CT^*, i.e., $CT = [CT_w, CT^*]$. To avoid any confusion, we stress that $CT^* = [T1,\ C1,\ C2,\ \{C1_y,\ C2_y \mid y \in Y1\}, \{C3_z,\ C4_z,\ C5_z \mid z \in Y2\}]$, where $Y1$ is the set of positive static attributes and $Y2$ is the set of negative static attributes.

Then, the encryptor X re-encrypts CT_w according to the dynamic attributes' access tree, $T2$. This is accomplished by calling the function $Encryption(CT_w, PK, T2)$ to generate the ciphertext, CT'_w. Note that the public key for both of the access trees are the same. The other part of CT, i.e., CT^* is not re-encrypted. The final ciphertext is $[CT'_w, CT^*]$.

To decrypt this ciphertext the receiver Y needs to have private keys for the leaf nodes of both of the access trees. A node can be given the corresponding private key related to the static attributes tree, $T1$ at the boot-up time. A more challenging question is to determine how the private keys related to the dynamic attributes can be distributed. We discuss how this can be done using the 'location' at-

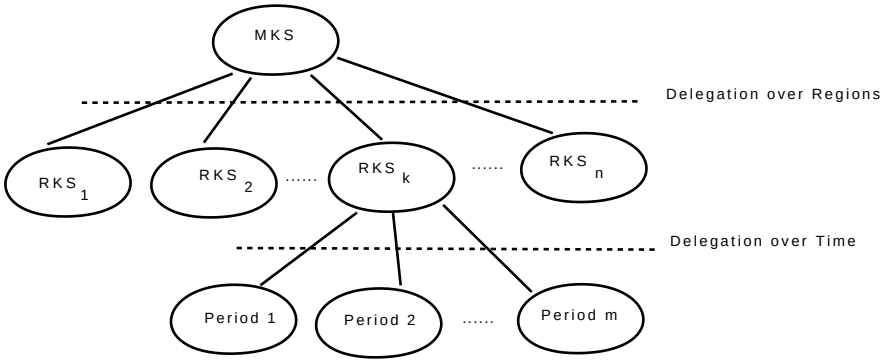

Fig. 3. Hierarchy used for private key distribution corresponding to location attributes: The Mobile Key Server (MKS) delegates the private key credential to the regional key servers (RKS_k) which, in turn, send the credential to the local users over several time periods.

tribute as an example. We assume that a node is given its private key corresponding to its current location by the regional key server (RKS_k) after the node proves its identity and presence to RKS_k. We exploit the 'delegation' property of CP-ABE to generate the private keys as shown in Figure 3. Let there be n regions in the whole network. We assume that in each region there is an RKS, which has the responsibility of dynamic private key distribution in its region. From the MKS, each RKS_k receives the private key for all of the attributes (i.e. for all of the time periods) corresponding to *Region* k. Using the delegation property, a RKS gives each local user the corresponding fresh private key once in each time period.

An Example: Let us discuss an example which illustrates how our scheme handles static and dynamic attributes. This example involves a battlefield scenario where US Army and Allied Force are deployed to counter an enemy force. The deployed participants may move from one location to another location and form a DTN via their wireless devices. In the perspective of US Army and Allied Force, the battlefield is divided into multiple strategic regions, i.e., '*Region* 1', '*Region* 2', and so on. Say the HQ wants to send a confidential message only to a few specific participants where the constraint is that the receiver has to be either a 'captain' of Allied Force or a '1st lieutenant' in US Army enrolled in 'Mission 3'. An extra condition is that the receiver has to be present in '*Region* k'. So, the access tree is as follows: (('captain' AND 'Allied Force') OR ('1st lieutenant' AND 'US Army' AND 'Mission 3')) AND ('*Region* k' AND '*Period* j'), assuming that the current time period is '*Period* j'. To achieve this access policy, we divide the access tree in two parts and apply encryption twice as discussed before. Following the above notations, the static access tree is $T1 =$ (('captain' AND 'Allied Force') OR ('1st lieutenant' AND 'US Army' AND 'Mission 3')), and the dynamic access tree is $T2 =$ ('*Region* k' AND '*Period* j'). The intended receivers obtain the necessary private keys corresponding to dynamic attributes from the regional key server RKS_k.

7 Conclusion

We have presented a CP-ABE scheme which can be used in DTNs. To introduce revocation facility in our scheme, we have adapted Bethencourt et al.'s [1] construction to accommodate the non-monotonic access structure proposed by Ostrovsky et al. [13]. We have further extended our CP-ABE construction to address the scenario where both static and dynamic attributes are present. There are several issues that warrant further studies. In this work, we assume that the data object is encrypted using a symmetric key which is later encrypted using the CP-ABE approach. Another approach is to encrypt the symmetric key using a key encryption key (KEK) and later encrypt the KEK using the CP-ABE approach. One may want to compare the pros and cons of using the KEKs. Furthermore, we also assume that each user can revoke any user using the negative attribute corresponding to that user's identifier. It is assumed that any user who revokes another user notifies the central authority so that the central authority can compile a list of revoked users periodically and disseminate such information to the regional key servers. Security analysis of such an approach needs to be studied more carefully.

Acknowledgments This work has been supported by DARPA under contract W15P7T-06-C-P430. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsor of this work.

References

1. J. Bethencourt and others. Ciphertext-policy attribute-based encryption. In *Proceedings of IEEE SP, Oakland*, 2007.
2. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *SIAM Journal of Computing*, 2003.
3. J. Burgess and others. Maxprop: Routing for vehicle-based disruption tolerant networks. In *Proceedings of IEEE Infocom*, 2006.

4. M. Chuah and others. Enhanced disruption and fault tolerant network architecture for bundle delivery (EDIFY). In *Proceedings of IEEE Globecom*, 2005.
5. M. Chuah and others. Node density-based adaptive routing scheme for disruption tolerant networks. In *Proceedings of IEEE Milcom*, 2006.
6. M. Chuah and others. Performance evaluation of content-based information retrieval schemes for dtns. In *Proceedings of IEEE Milcom*, 2007.
7. M. K. et al. Plutus: scalable secure file sharing on untrusted storage. In *Proceedings of ACM Usenix*, 2002.
8. V. Goyal and others. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of ACM CCS*, 2006.
9. A. Harrinton and C. Jensen. Cryptographic access control in a distributed file system. In *Proceedings of ACM SACMAT*, 2003.
10. Lingren and others. Probabilistic routing in intermittently connected networks. In *Proceedings of Workshop on Service Assurance with Partial and Intermittent Resources*, 2004.
11. B. Lynn. The pairing-based cryptography (pbc) library. In *http://crypto.stanford.edu/pbc*.
12. McDaniel and others. A response to "can we eliminate certifacte revocation lists?". In *Proceedings of FC*, 2000.
13. R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, 2007.
14. M. M. B. Tariq and others. Message ferry route design for sparse ad hoc networks with mobile nodes. In *Proceedings of ACM Mobihoc*, 2006.
15. S. Yu, K. Ren, and W. Lou. Attribute-based content distribution with hidden policy. In *Proceedings of the Fourth Workshop on Secure Network Protocols (NPSEC)*, 2005.