# Secure Descriptive Message Dissemination in DTNs

M. Chuah
CSE Dept
Lehigh University
Bethlehem, PA 18015, USA
chuah@cse.lehigh.edu

S. Roy
CSE Dept
Lehigh University
Bethlehem, PA18015, USA
ros308@cse.lehigh.edu

I. Stoev
CSE Dept
Lehigh University
Bethlehem, PA 18015, USA
ivs209@lehigh.edu

## 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. A key feature of DTN technology is the late-binding capability that allows messages destined to a descriptive name to be resolved progressively until the messages are delivered to one or several recipients. In this paper, we describe a flexible security solution that allows messages destined to descriptive names to be sent securely such that unauthorized personnel is not allowed to eavesdrop on them. Our solution builds on the attributed based cryptography system. In addition, we describe our implementation of a late-binding router that supports our security solution. We also present our prototyping experience.

## Categories and Subject Descriptors
C.2.0 [**Computer-Communication-Networks**]: General; C.2.2 [**Computer-Communication-Networks**]: Network Protocols

## General Terms
Design, Security

## Keywords
Security, Descriptive Message Delivery, Disruption Tolerant Networks

## 1. INTRODUCTION
With the advancement in technology, many users carry small computing devices e.g. PDAs, cell-phones etc with wireless interfaces. These devices can form mobile ad hoc networks and communicate with one another via the help of intermediate nodes. Such ad hoc networks are very useful in several scenarios e.g. battlefield operations, vehicular ad hoc networks and disaster response scenarios. Many ad hoc routing schemes have been designed for ad hoc networks but such routing schemes are not useful in some challenging network scenarios where the nodes have intermittent connectivity and suffer frequent partitioning. Recently, disruption tolerant network (DTN) technologies [1],[2] have been proposed to allow nodes in such extreme networking environment to communicate with one another. New routing schemes more appropriate for DTN environment are also proposed e.g. [17]. Three key ideas in the DTN architecture are (a) intentional naming, (b) late binding and (c) persistent delivery. Intentional naming [1,4,5] means that nodes can be addressed using some descriptive attributes of the roles of their users e.g. all mall shoppers in the 2nd floor. It is often useful to be able to address nodes using both the standard DTN endpoint identifier (EID)s as well as some descriptive attributes of nodes or users of nodes. A similar idea has been proposed in MIT's intentional name system [3]. However, the application of these ideas to DTNs is relatively new [1],[4]. Some examples of intentional naming are shown in Table 1.

**Table 1: Examples of Intentional Names**

| Category | Examples |
|---|---|
| Node-Attribute | All nodes equipped with GPS |
| | All nodes which detect poisonous gas |
| User-Attribute | All EE students at Lehigh |
| | All soldiers from Company A |
| Location Based | All vehicles within 200m of Bethlehem City Center |
| | All nodes within 500m of Manhattan Bridge |
| Combined User & Location Based | All medics within 200m of Allentown City Building |
| | All firefighters within 200m of Wall Street Building |
| | All CSE students within Packard Laboratory |

Late binding [4,5] means that the mapping of an intentional name to an address needs not happen when a packet (or referred to as a bundle in the DTN architecture) is created. In a conventional network, the mapping from a name to an address usually happens very early when packets are constructed and passed to the routing layer. In a DTN, due to the sparse connectivity, it may not be feasible for a source to have the necessary information to bind values of attributes (e.g. roles, locations) to addresses. Therefore, we need to have a process of progressively mapping an intentional name to canonical EIDs. This process is referred to as "intentional name resolution". Persistent delivery [5] means that bundles will be stored for further delivery at an intermediate node if that node can not yet resolve the intentional name of these bundles or find no routes yet to the resolved addresses of these bundles. Since DTN nodes persistently store information, bundles can be delivered to recipients that match the description but are not present at the time a bundle is received. Prototypes supporting DTN are available but only one [5] provides the intentional name and late binding features. The prototype in [5] is not available to the general public. In this work, we develop a late binding router implementation that supports these features and our prototype will be made available in the near future.

In some DTN application scenarios e.g. mobile services for


*MobiOpp'10*, Feb 22-23, 2010, Pisa, Italy.

vehicular ad hoc networks or military operations, the communications between peers need to be secured. Bundle security protocol (BSP) [14] has been defined to secure such communications. The bundle authentication block (BAB) feature provides a per hop source and bundle integrity check. The payload security block (PSB) feature provides an end-to-end source and payload integrity check. This is achieved via the public key signing of a SHA256 hash of the payload. The confidentiality block (CB) feature provides for the encryption of the entire bundle payload which is usually done at the source and decrypted at the final destination. The security features provided in BSP are not as flexible as what one would like to have. For example, one may form a secure communication group dynamically and hence merely relying on configuration rules (e.g. use Key1 for all CS faculty members) is not an attractive solution. In this paper, we propose a security solution that is based on a cipher-text attribute-based encryption (CP-ABE) scheme [9]. Our proposed solution contains several features that are not found in the basic CP-ABE scheme described in [9], namely (i) our solution supports negative attributes, (ii) our solution provides revocation feature, (iii) our solution supports dynamic attributes. In addition, we develop a prototype that incorporates our security solution into our late-binding router which provides resolutions of intentional names that are combinations of role and location-based attributes.

In summary, our contributions are (i) design an enhanced CP-A BE scheme with three new features that allow secure descriptive based messages to be sent, and (ii) develop a prototype that incorporates our security solution, intentional name resolution and late binding features. The rest of the paper is organized as follows: In Section 2, we discuss related work. In Section 3, we describe how the resolution of intentional naming and routing can be performed. Next, we describe our CP-ABE based security solution in Section 4. Last but not least, we describe in Section 5 how our security solution can be used for secure descriptive based message dissemination. We also describe the testbed experience we had with our preliminary implementation that provides both the late-binding and CP-ABE based security features. We conclude with some discussions about future work in Section 6.

## 2. RELATED WORK

In [4], the authors describe the SPINDLE disruption tolerant network system that they design. They describe a name management architecture that supports progressive resolution of intentional name attributes. Their name management architecture supports an expressive name scheme based on a descriptive logic language. They also describe two specific routing algorithms, namely the prioritized epidemic (PREP) and the anxiety-prone link state (APLS) schemes. In the PREP scheme, bundles that require fewer hops to reach the destination will be less likely to be dropped than bundles that require more hops. In addition, a higher transmission priority is given to those bundles that are heading "downstream" and also to bundles that have higher remaining time to expiration. The APLS scheme enhances the PREP scheme by introducing a shortest cost path mode that is triggered whenever the cost to the destination is less than a configured threshold. The paper briefly describes their name management architecture but provides no detailed descriptions to help others understand how it works. In [5], the authors provide more information on the syntax of the intentional names that their system can handle and also provide some descriptions on how their system performs intentional name resolution. However, their solution does not include any security features. We will discuss how our implementation of intentional name resolution differs from theirs in Section 3.

Some work has been done on providing secure group communications in vehicular ad hoc networks. For example, in CARAVAN [6], group formation is performed based on the assumption that all vehicles forming a group are moving at the same average speed while maintaining relatively constant distance from one another. Such an assumption does not hold in many ad hoc networking scenarios. In [7], the authors proposed a group signature scheme by grouping vehicles based on their region and role. However, the group communication among vehicles from different categories is constrained. In [8], the authors propose using an ABE based scheme for policy-based group key management and suggest using different access trees for dynamic attributes but do not provide much details of their constructions. They also do not describe any revocation feature.

## 3. ROUTING WITH INTENTIONAL NAMES

In this paper, we assume that the DTN EIDs follow URI syntax (scheme: scheme-specific-part) [2]. The scheme-specific part (SSP) of a DTN intentional name should be capable of denoting an individual DTN endpoint (node or a service) or a group of nodes, with potentially multiple attributes. In the scheme-specific part, we follow a convention similar to what BBN proposes in [5] e.g. dtn:intent#(role(student), loc((1,3), 100m) means that we want to reach all students within 100m of a location with coordinates (1,3). To avoid ambiguity, we also follow the recommendation suggested by BBN that an ontology name should be put in front e.g. Lehigh: role(student). Here, the term 'Lehigh' refers to the Lehigh ontology that contains the definitions of rules such as 'role'.

In our work, we assume that each node has an ontology definition. In cases where the nodes may not have a copy of the ontology definition, a node can be selected as the resolver and other nodes can send queries to this resolver for name resolution or request for a copy of the ontology definition. To avoid high control message overhead, the resolver may also publish its ontology rules using a hop-count based dissemination scheme.

The steps taken to resolve an intentional name are similar to what is described in [5]. When a node is presented with an intentional name, the following steps need to be performed:

1. if an intentional name can be completely resolved to a canonical EID, the resolution task is completed.
2. if the name can be partially resolved/simplified, the node will do so.
3. if the node knows of any nodes that can resolve the name further, then it forwards the bundle to such nodes.
4. if none of the above applies, then the node cannot perform any further resolution. Then, it follows a default behavior, e.g. store the bundle until a fresh name update occurs or flood the bundle. The default behavior can be configured.

The following scenario is a more concrete example of the resolution process: let say a track event organizer wishes to send a message to all advisors within 100m of Building A. The intentional name looks as follows: dtn:intent#(Role('advisor') && Loc('Building A', 100m)). Since the source does not know the EID(s) of the advisors that satisfy the above location constraints, it is unlikely that the source can perform early binding. In order to resolve the name, the source first determines the location of Building A, route the bundle to nodes within 100m of that location. If the source does not have any geographical information of Building A, then the initial resolution of the geographical portion of the intentional name can be performed at any node that the source is aware of which has geographic information. This would result in a physical location but still no canonical EIDs yet. Next, the source attempts to select the next hop care-of node that is at or near the target region. The source queries its node location database, finds a node close to the target region, selects that as the care-of node, and routes the bundle to it.

Let us assume that the chosen care-of-node is located within the target region, and that it has radio contacts with all nodes in the target region. Upon receiving the routed bundle, that care-of-node will attempt to resolve the first part of the intentional name, i.e. Role('advisor'). If the care-of node does not have such information, it will re-broadcast the bundle. Each receiving node compares its role, and either accepts or discards the bundle. Alternatively, if the care-of node has information about the 'roles' of nearby nodes, it can compile a list of the canonical EIDs of relevant nodes and send the bundle to these nodes individually.

**Interaction with DTN routing**

We describe how the late binding feature fits into the DTN2 reference implementation. The steps for the interaction between routing and late binding modules are outlined in Figure 1. The core bundle protocol agent (BPA) which is responsible for implementing the bundle protocol and the bundle security protocol, interacts with other decision plane modules (Prophet/Epidemic/RAPID routers and Resolver), convergence layer adaptor (CLA), DTN application (APP), and data storage (DS).

**Dissemination:** Our late binding (LB) router consists of two modules, namely a routing decision module, and a resolver module. An application registers itself with BPA and provides intentional attributes and values. The attributes are installed in the local name knowledge base (KB). A node broadcasts its location periodically. Upon receiving the location from another node, the LB router inserts/updates the location information of neighboring nodes into its local KB. The main differences between our implementation and BBN [5] are: (i) the bundle receive event notifications are sent to only the router module, router parses event/control messages and sends queries to the resolver while both modules parse the meta-data information in BBN's approach, (ii) we develop codes for our own reasoner while BBN used a more sophisticated inference engine off the shelf e.g. XSB [20] which does not give good performance.

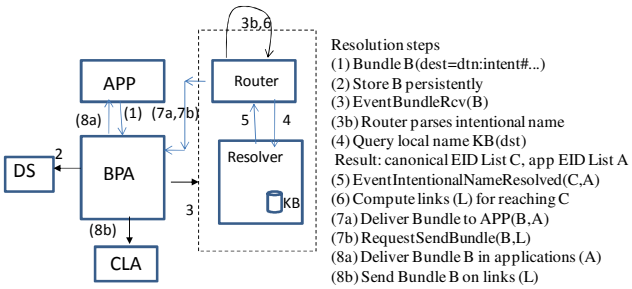



Figure 1: Resolution Steps

**Resolution**: The resolution steps are shown in Fig 1. When a bundle arrives, it is first stored in the local data store, and an EventBundleReceived event is triggered. This event is handled by the local router. The router parses the intentional name, queries the resolver, and is returned one or more canonical EIDs for next-hop care-of-nodes. The router then determines the appropriate links to reach these nodes. It is possible that a resolver's name KB does not have enough information to resolve a name immediately, but an unresolvable name may become resolvable when the router receives some new information. Thus, bundles with unresolved intentional names are stored in a special queue. Whenever new information is available e.g. hearing a beacon from a new node, then the resolver of that node will attempt resolution again. If the resolver can complete the task, the successful name resolution result will be provided to the router. The router then finds an existing link to send the bundle out.

We do not know the details of how intentional names are resolved in BBN's implementation. In our implementation, if the intentional name contains a location predicate, then the location predicate is processed first. If the current node's location is within the target destination region of the bundle, then the router will set the dissemination mode to "flood". Otherwise, the router finds the best next-hop node to forward the bundle. If the intentional name does not contain a location predicate, then all the predicates in the intentional name are sent to the resolver module. The resolver queries its knowledge base for canonical EIDs that satisfy the criteria given by the predicates. The list of canonical EIDs are returned to the local router and the local router will determine the links that can be used to reach these canonical EIDs. Bundles are then sent to these links. Bundles are stored in persistent storage so that they can be delivered to nodes that are encountered later. Periodically, stored bundles which have expired are purged from the persistent storage.

# 4. OVERVIEW OF CP-ABE SOLUTION

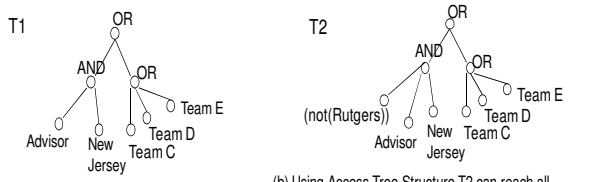
Since some messages should only be interpreted by the final recipients who have the authorization to do so, and the sender may not know who the recipients are when his message is constructed, an encryption scheme that allows messages to be encrypted based on conjunctions and disjunctions of attributes should be provided. In this work, we use an enhanced ciphertext attribute-based encryption solution. In subsequent subsections, we first describe the basic CP-ABE scheme and later describe the enhanced features we provide.

## 4.1 Basic CP-ABE scheme

Attribute-based ABE is an access control mechanism proposed by Sahai and Walters [8]. In ABE scheme, the encryptor creates an

access tree structure *T* with a set of attributes and threshold gates. The access tree structure describes the access control policies that a user must satisfy to decrypt a particular message. To decrypt the message, the user must own the secret keys associated with the access tree structures over the set of attributes. These secret keys are generated by a trusted authority.

Ciphertext Policy Attribute Based Encryption (CP-ABE) [9] scheme extends ABE by allowing arbitrary access formulas in the ciphertext policy. CPABE scheme utilizes identity based schemes (IBE) [10] and threshold secret sharing scheme [11]. In a CP-ABE system, each user is associated with a set of attributes. When encrypting a message *M*, the encryptor specifies an access tree 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. Let us consider a sport event scenario where track teams from various schools in the tri-state area meet. The teams are given names such as Team A, B, etc. Each team has an advisor. Assume that the organizer wants to send a message to all advisors from New Jersey or anyone that is a team member of Teams C, D, and E, then the access tree structure will look like Fig 2(a).




(a) Using Access Tree Structure T1 can reach all team members of Teams C,D, E and all advisors from New Jersey

(b) Using Access Tree Structure T2 can reach all team members of Teams C,D, E and all advisors from New Jersey except Rutger's advisor

**Fig 2: Access Tree Structures without/with Negative Attribute Revocation Feature**

A CP-ABE scheme consists of four algorithms:

**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.

**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 $S_X$ .

**Encrypt:** This is a randomized algorithm that takes as input a message *M,* an access structure *T,* and the public parameter *PK*. It outputs the ciphertext *CT*.

**Decrypt:** This algorithm takes as input the ciphertext *CT* and a secret key *SK* for an associated attribute set $S_X$. Only if $S_X$ satisfies the access structure embedded in *CT* will it return the message *M*.

## 4.2 Enhanced CP-ABE scheme

Our CP-ABE solution provides three enhanced features over the basic scheme described in [9]. First, the basic protocol does not support negative attributes i.e. the ability to include an access control feature such as (not(Team 5)). Using, the earlier track team example, let us assume that the organizer does not want his message to go to the advisor from Rutgers, then the access structure shown in Fig 2(b) will be used. This access structure uses a negative attribute, namely (Not Rutgers).

We modify the basic CP-ABE construction to accommodate a non-monotonic access structure such that negative attributes can be supported. Second, the basic scheme does not provide user-based revocation feature but our solution does. Our revocation feature makes use of the negative attribute feature we added. Third, our solution allows for dynamic attributes to be included. We support dynamic attribute by using two "integrated" access trees: one is for static attributes with a dummy leaf node while the other is for dynamic attributes. When the second tree is satisfied, a user can obtain the private key component associated with the dummy leaf node present in the first access tree. We discuss more details in subsequent paragraphs.

We first briefly sketch the construction of our solution that allows us to support negative attributes. Details can be found in [16]. To describe our construction, we use the same terminology and language as in [9,10]. In our construction, each leaf node of the access tree *T* represents either a positive or negative attribute. An example of a positive attribute is "Advisor" and an example of a negative attribute is "Not Rutgers". However, private key components are only assigned to positive attributes, i.e. for any user X, $S_X$ does not contain any negative attribute. Each internal node of the access tree *T* represents a threshold gate, which can be an "AND" gate or an "OR" gate. If $num_x$ is the number of children of a node, x, and $k_x$ is its threshold value, then $0 < k_x \le num_x$. We denote the parent of node x by parent(x). 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 further map each attribute to a unique integer in $Z_p^*$, e.g. using a collision hash function F: $\{0,1\}^*$-> $Z_p^*$. For example, let say the attribute of a leaf node x is "Advisor", then att(x)=F('Advisor') $\in Z_p^*$. Furthermore, a function *H:* $Z_p$-> $G_0$ is used to map any attribute, *x*, to an element in $G_0$ where $H(i)=g^i$ where *att(x)=i*. Our cryptosystem consists of the following algorithms:

**Setup:** This algorithm chooses three random exponents $\alpha, \beta, \gamma \in Z_p$. A parameter *d* specifies how many attributes this system use. A polynomial *v(x)* of degree *d* is chosen at random subject to the constraint that *v(0)=β.* The public parameters are as follows:

$$PK = [G_0, g, h = g^{\beta}, h^{\gamma}, f = g^{1/\gamma}, e(h,g)^{\alpha}, \{g^{v(0)}, g^{v(1)}, ..g^{v(d)}\}]$$

The master key *MK* is [ $\beta, \gamma, g^{\alpha}$ ]. Note that PK is public information shared with all users.

**KenGen(*MK,S*)** This algorithm takes as input a set of attributes S and outputs a secret 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 secret key SK as

$$SK = [D = g^{(\alpha + r)/\gamma}, D_1 = g^r \forall j \in S, D1_j = h^r . H(j)^{r_j}, D2_j = g^{r_j}, D3_j = (V(j))^r]$$

**Delegate(*SK*, $\overline{S}$ )** The delegation algorithm takes in a secret key SK which is for a set S of attributes, and another set $\overline{S}$ such that

$\overline{S} \in S$. The algorithm chooses random $\overline{r}$ and $\overline{r}_k \forall k \in \overline{S}$. Then, it creates a new secret key as

$\overline{SK} = [\overline{D} = D.f^{\overline{r}}, \overline{D}1 = g^{r}.g^{\overline{r}}, \forall k \in S, \overline{D}1_k = D1_k.h^{\overline{r}}.H(k)^{\overline{r}_k},$
$\overline{D}2_k = D2_k.g^{\overline{r}_k}, \overline{D}3_k = D3_k(V(k))^{\overline{r}}]$

**Encryption (*PK, M, T*)** The encryption algorithm encrypts a message *M* under the tree access structure *T*. As in the basic scheme, this algorithm first chooses a polynomial $q_x$ (with degree $d_x=k_x-1$) for each node *x* (including the leaves) in the tree T. For example, at the root node R of the access tree, 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. Let *Y1* and *Y2* be the set of the leaf nodes in *T* with positive and negative attributes, respectively. For each node $y \in Y2$, we randomly choose $u_y$ from $Z_p$. Recalls that the function *F* maps each attribute into an element in $Z_p^*$. A function $V(i):Z_p \rightarrow G_0$ is defined as $V(i) = g^{v(i)}$. Then, the ciphertext *CT* is constructed as follows:

$$CT = [T, C1 = Me(h,g)^{\alpha s}, C2 = (h^{\gamma})^{s},$$

$$\forall \text{ nodes } y \in Y1 : C1_y = g^{q_y(0)}, C2_y = H(i)^{q_y(0)}, \text{where } i = F(att(y)), \text{and}$$

$\forall$ *nodes* $y \in Y2 : C3_y = h^{q_y(0)+u_y}, C4_y = (V(i))^{u_y}, C5_y = g^{u_y}$ where $i = F(att(y))]$. (Note that these are not in the basic scheme)

**Decrypt(*CT,SK*):** First, we define a recursive algorithm *DecryptNode(CT,SK,x)* that takes as input a ciphertext CT, a secret key SK, which is associated with a set of attributes S, and a node x from the access tree *T*. If the node *x* is a leaf node, then we let *i=att(x)*. If *x* corresponds to a positive attribute, and $i \in S$, then

$$DecryptNode(CT, SK, x) = \frac{e(D1_i, C1_x)}{e(D2_i, C2_x)}$$
$$= \frac{e(h^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} = \frac{e(h^r, g^{q_x(0)}) \cdot e(H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})}$$
$$= e(h^r, g^{q_x(0)}) = e(h,g)^{rq_x(0)}$$

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

When a leaf node *x* is satisfied by *S*, then *DecryptNode(CT,SK,x)*= $e(h,g)^{rq_x(0)}$. Interested readers can refer to [16] for details.

Next, we can 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 in $L_c$. Let $U_x$ be an arbitrary $k_x$ sized set of child nodes such that $L_c$ is not an empty set. If no such set exists, then the node *x* was not satisfied by S and *DecryptNode(CT,SK,x)*= $\perp$. Otherwise, we can verify that *DecryptNode(CT,SK,x)*= $e(h,g)^{rq_x(0)}$. Thus, the decryption algorithm begins by simply calling the function on the root node *R* of the access tree *T*. If the tree is satisfied by *S*, then we set A= *DecryptNode(CT,SK,R)*= $e(h,g)^{rq_R(0)} = e(h,g)^{rs}$. Then, the decryption algorithm can decrypt the ciphertext by computing

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

Since the CP-ABE solution can be expensive in terms of computations, our security solution combines the symmetric key solution with our enhanced CP-ABE solution. In our security solution, each data publisher encrypts his data items using symmetric keys. The symmetric keys are then encrypted using our enhanced CP-ABE scheme such that only authorized personnel can decrypt these messages to retrieve the symmetric keys, and then use these symmetric keys to decrypt the encrypted data items.

**Supporting User Revocation**

A simple solution for the revocation feature is to include a time attribute. This solution requires 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, e.g. $T'=(T \text{ AND } \theta)$ where $\theta$ can be the date attribute. It is assumed that each non-revoked user receives his fresh private keys corresponding to the date attribute once everyday directly from a mobile key server. Apart from this inconvenience, this simple revocation solution is a lazy revocation technique since the revoked user is not purged from the system until the current time slice expires.

To address these issues, we propose a more practical solution. We assume that each user has an attribute that represents its own ID. Then, we can revoke a user using our enhanced CP-ABE scheme which allows negative attributes in the access policy. For example, if a message is sent to all team members of Team A except Bob, we can use an access tree that says "AND (NOT('Bob'), Team A)". We assume that regional key servers can periodically inform nearby users with a list of revoked users. Note that this is not inconsistent with the late binding feature since in most scenarios, we would like to invoke a particular individual.

**Handling Dynamic Attributes**

The attributes of a user may be static (meaning the attribute value does not change for a long time) or dynamic (meaning its value may change frequently e.g. location). A node may be in Region *K* at time t1 and Region *L* at time t2. A sender may only want nodes in Region *L* to be able to decrypt his message. Thus, we need to address the issue of letting an authorized user *X* decrypt a message that has been encrypted using an access tree that has both static and dynamic attributes. A user *X* needs to be furnished with the private key component corresponding to the current value of a dynamic attribute. We assume that once a user qualifies for a new value of a dynamic attribute e.g. a user has entered into a new region and has authenticated itself with a regional key server that is in charged of that region, then, that user can get the relevant private key component from the associated regional key server. To deal with dynamic attributes, we propose the following solution.

In our proposed solution, we assume that the access policy can be written in such a way that the set of static and dynamic attributes

can be connected by an "AND" gate. For example, ('*Team A*' OR '*Team B*') AND ('Region *K*')). Then, we can divide the access structure into two distinct parts – one tree *T1* with the static attributes and another tree *T2* with the dynamic attributes as shown in Fig 3. Similar idea has been proposed in [18] but the authors do not provide much details or build any prototype.

In *T1,* we introduce a dummy leaf node *w* which represents a dummy attribute on behalf of the whole set of dynamic attributes 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* performs the function *Encryption(M,PK,T1)* to generate the ciphertext *CT*. From the encryption algorithm described in Section 4.2, we see that there is a component in *CT* which corresponds to the leaf node *w* in *T1*. Let us refer to this component as $CT_w$ which is $[C1_w, C2_w]$. The rest of *CT* is referred to as $CT^*$ i.e. $CT=[CT^*, CT_w]$ where $CT^*=[T1, C1, C2, \{C1_y, C2_y \mid y \in Y1\}, \{C3_z, C4_z, C5_z \mid z \in Y2\}]$, *Y1* is the set of positive static attributes, *Y2* is the set of negative static attributes. Then, the encryptor 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 $CT_w^{'}$. The final ciphertext is now $[CT^*, CT_w^{'}]$.

To decrypt, the receiver *Y* needs to have private key components for the leaf nodes of both access trees. A receiving node can be given the corresponding private key components related to the static attributes at boot-up time and then obtain the private key components for the dynamic attributes from a relevant regional key server if these dynamic attributes are associated with location. The receiving node should authenticate itself with the regional key server before the regional key server can distribute private key components associated with the dynamic attributes to this receiving node. Note that the delegate feature may be used to allow the regional key server to change the private key components associated with locations periodically so that users who have received these key components cannot decrypt future messages once their private key components expire.

# 5. SECURE DESCRIPTIVE MESSAGE DISSEMINATION

Our proposed secure description-based message dissemination system uses the intentional naming system, and the enhanced ciphertext attribute encryption scheme described in Sections 3 and 4. We first use the example shown in Fig 2 to illustrate how we make use of the intentional name and enhanced CP-ABE features to send secure descriptive messages. Let us assume that the organizer wants to send secure messages to all players from New Jersey currently located 100m away from Building A. Then, he uses the following intentional name: dtn:intent#(State='New Jersey' AND within(Building A, 100m)). The message is encrypted using two attributes – a location-based, and a state-based attribute. The organizer's node sends this message to the nearest care-of node found within the destination region (which is 100m from Building A) from its local knowledge base. That care-of node will either (a) derive a list of DTN canonical EIDs of all New Jersey players currently in that target region which its local resolver is aware of, and sends an individual message to each one of them, or (b) re-broadcast the message so that any node with matching attributes will receive, and decrypt successfully the encrypted message.

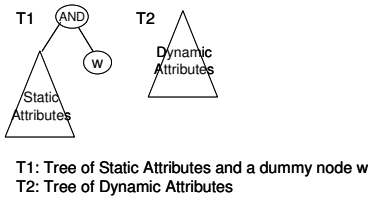



**Figure 3: Access tree with static/dynamic attributes**

Our late binding implementation was built on top of the Java-based RAPID router [17]. RAPID router is implemented as an external router that communicates to the DTN2 layer via several APIs. We use the reference DTN2 implementation version 2.5 in our work. The reference DTN2 implementation [12] complies with the bundle protocol specifications described in [13]. It also supports basic bundle security protocol features described in [15]. It does not support the MEB extension discussed in [14]. We modify the router code of the RAPID router to parse intentional name, insert and update local knowledgebase. We also create a resolver function that can search the local knowledge base for canonical EIDs that match certain predicates. There are some limitations in our current implementation:

1. Currently, only location updates are shared with neighboring nodes. More information from the local KB can be shared with neighboring nodes periodically, e.g. if every node shares information of its 1 hop neighbors periodically, a 2-hop neighborhood can be derived
2. Our LateBinding Router evaluates location predicates first. This is done to minimize the network resource usage before the bundle reaches its target destination region. There may be special cases where resolving other attributes first may result in smaller network resource consumption. We leave other resolving strategies for future work.
3. When a location predicate is present in the intentional name, the bundle is only forwarded to a care-of-node that is closer to the target destination region. For reliability purpose, one may use another geographical forwarding policy e.g. choosing all neighboring nodes that are closer to the target destination region than the current node.

Our enhanced CP-ABE scheme is implemented using the PBC-library [21]. We first run our security implementation on an Intel Core TM) 2 Quad 2.4 GHz workstation. With d=20 (attributes), the average time (over 30 runs) taken for the Setup algorithm is 0.076 sec while the encryption algorithm takes an average of 1.332 sec with 15 positive attributes and 10 negative attributes. The decryption takes an average of 1.98 sec.

Next, we set up a test topology shown in Fig 4 to demonstrate our secured descriptive message dissemination solution. Node n2 sends a message with the following access tree, T=(Role('advisor') && Loc(X,Y,R)) where the target destination region is a circular area located within a radius of R m from a centered location with coordinate values (X,Y). Let assume only nodes n3 and node 5 have roles as 'advisor'. Node n2 determines that node n7 is closer to the target destination region, and hence forwards the bundle to node n7. Similarly, node n7 (n1) determines that node n1 (n4) is closer to the target destination region and hence forwards the encrypted bundle to node n1 (n4).

Finally, node n4 determines that it is within the target destination region and hence changes the dissemination mode to "flood". Thus, the bundle is flooded to all the links which n4 is aware of (namely to nodes n3, n6). Node n6 determines that this message needs to be further relayed to node n5. Node n5 determines that it has attributes that satisfy the access tree of this encrypted bundle. Finally, node 5 decrypts the message and sends it up to the application layer. Similarly, node n3 determines that this encrypted bundle is meant for itself, decrypts and sends it up to the application layer. Fig 5 shows the image embedded within the encrypted bundle which nodes 3 & 5 successfully decrypt.

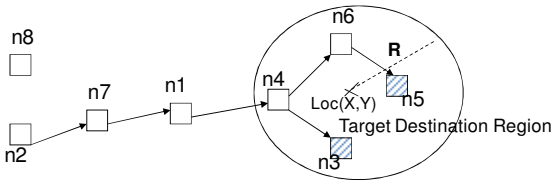



**Fig 4: Testbed Topology**

## 6. CONCLUDING REMARKS

In this paper, we have proposed a secure descriptive message delivery system which provides both (a) the late binding feature (i.e. bundles need not be bound to canonical EIDs when they are generated) feature, and (b) access control feature that is based on an enhanced ciphertext based attribute encryption scheme. We have also described an implementation of our solution that is built on top of the RAPID router and the reference DTN2 implementation. There are some limitations in our current limitations. In the near future, we hope to enhance our implementation to remove these limitations. We intend to make use of the meta-data extension [14]. In addition, we hope to explore another more scalable user revocation approach using broadcast encryption [19].

## 7. 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. We wish to thank Dr. Peng Yang for his initial work on the prototyping of our late-binding router and B. Herbst for his help in the prototyping work.

## 8. REFERENCES

[1] K. Fall, " A delay tolerant network architecture for challenged networks", Proceedings of ACM Sigcomm, 2003.

[2] V. Cerf et al, "Delay Tolerant Networking Architecture", RFC4838, April 2007

[3] W Adjie-Winoto, E. Schwartz, H. Balakrishnan, "The design and implementation of an intentional naming system", Proceedings of ACM SOSP, Dec 1999

[4] R. Krishnan et al, "The SPINDLE Disruption Tolerant Networking System", Proceedings of IEEE Milcom, 2007

[5] P. Basu, R. Krishnan, D. W. Brown, "Persistent Delivery with Deferred Binding to Descriptively Named Destinations", Proceedings of IEEE Milcom, 2008.

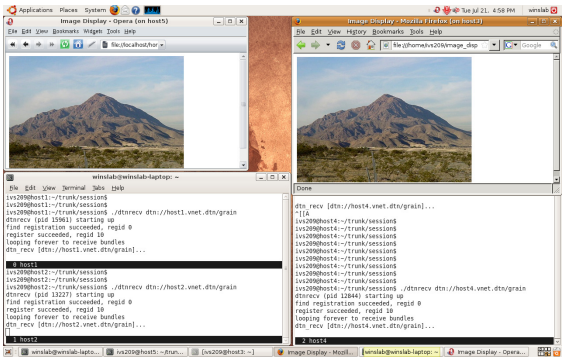


**Figure 5: Nodes n3 and n5 successfully decrypt the encrypted bundle**

[6] K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Masurra, K. Sezaki, "CARAVAN – providing location privacy for VANET ", Proceedings of Embedded Security in CARs (ESCAR), 2005.

[7] J. Guo, J. P. Baugh, S. Wang, "A group signature based secure and privacy preserving vehicular communication framework", Proceeding of Mobile Networking for Vehicular Environments (MOVE), pp 103-108, 2007.

[8] V.Goyal et al, "Attribute-Based Encryption for fine-grained access control of encrypted data", Proceedings of ACM CCS, pp 89-98, 2006

[9] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-policy attribute-based encryption", Proceedings of 28th IEEE Sympoisum on Security and Privacy, 2007

[10] D. Boneh, M. Franklin, "Identity Based Encryption from the weil pairing", Proceedings of Crypto 01, Springer-Verlag, 2001

[11] A. Shamir, "How to share a secret", Communications of the ACM, 22(11),pp 612-613,1979

[12] M. Demmer, "DTN Reference Implementation, Version 2", URL http://www.dtnrg.org

[13] K. Scott, S. Burleigh, "Bundle Protocol Specification", RFC 5050, Nov, 2007

[14] S. Symington, "Delay-Tolerant Networking Metadata Extension Block", draft-symington=-dtnrg-bundle-metadata-blcok-01, work in progress, Feb, 2008

[15] S. Symington, F Farrell, H. Weiss, P. Lovell, "Bundle Security Protocol Specification", draft-irtf-dtnrg-bundle-security-05 (work in progress), Feb, 2008.

[16] S. Roy, M. Chuah, "Secure Data Retrieval Based on Ciphertext-Based Attribute Based Encryption System for Disruption Tolerant Networks", Lehigh CSE Technical Report, May, 2009. http://www.cse.lehigh.edu/~chuah/publications/cpabe_report09.pdf

[17] A.Balasubramanian, B Levine, A. Venkataramani, "DTN Routing as a Resource Allocation Problem", Proceedings of ACM Sigcomm, Aug, 2007.

[18] X. Hong, D. Huang, M. Gerla, Z. Gao, "SAT: situation-aware trust architecture for vehicular networks", Proceedings of ACM Mobiarch, 2008.

[19] D. Boneh, C. Gentry, B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys", Proceedings of Crypto, 2005, LNCS 3621, pp 258-275, 2005.

[20] SUNY Stony Brook, "XSB Logic Programming and Deductive Data Base System", http://xsb.sourceforge.net.

[21] B. Lynn. The pairing-based library http://crypto.stanford.edu/pbc