Secure Data Retrieval System (SEDAR) for DTNs

M. Chuah, R. Metzger

Department of Computer Science & Engineering
Lehigh University
chuah@cse.lehigh.edu, rcm2@lehigh.edu

Abstract—Disruption tolerant networking (DTN) technology is designed to deal with the intermittent connectivity among mobile nodes due to mobility, short range radios or terrain obstacles. We have recently evaluated the performance of two information retrieval schemes that are designed to work in the DTN environments. Our schemes utilize both query and data replications to enhance the query success rate. Besides designing efficient query and data dissemination schemes, one needs to consider the security aspects since sensitive data should only be accessed by authorized personnel. In this paper, we present a data-centric security solution for an information retrieval system which we design for DTN environments. We also describe the preliminary prototype that we have built.

Keywords—*disruption tolerant networking; data-centric; security; prototype*

I. INTRODUCTION

With technology advancement, many computing devices e.g. PDAs, smart-phones, sensors have wireless interfaces and hence can form ad hoc networks. Wireless adhoc networks allow nodes to communicate with one another without relying on any fixed infrastructure. These rapidly deployable networks are very useful in several scenarios e.g. battlefield operations, disaster relief centers etc. Routing algorithms designed for adhoc networks, e.g. [1], do not work in challenging network scenarios where node mobility, terrain obstacles, limited radio range, etc often result in frequent network partitions.

Recently, some solutions have been proposed to deal with such challenging communication scenarios. For example, disruption tolerant network (DTN)s [4],[5] are designed to overcome such limitations. Several routing schemes have been proposed for DTNs. Some use history based information to estimate delivery probability of peers and pass the messages to the peer that can best deliver the messages [7],[8]. Such routing schemes rely on the node mobility to deliver packets amidst frequent network partitions using a store and forward approach.

Although routing is an important design issue, the ability to access information rapidly is also an important feature that a DTN should provide since the ultimate goal of having

such a network is to allow mobile nodes to access information quickly and efficiently. For example, in a battlefield, soldiers need to access relevant information e.g. terrain descriptions, weather, intelligent information, locations of enemy and friendly forces etc. In [3],[9], we have designed two information retrieval schemes that use query/data duplications to enhance the query success rate in DTN environments. Our results show that the scheme that binary spreads replicated data copies and queries can achieve 45% to 460% higher query success ratio when compared to a scheme that does not use any data and query replication. In [3],[9], we do not address any security design.

In this paper, we describe a secure mobile data retrieval system designed for disruption tolerant networks. Specifically, we present the system model, the publish/subscribe model, the data retrieval schemes and a data-centric security solution which we design. In our data-centric security solution, authorized users derive encryption keys for the data items which they are authorized to access from the access keys they get when they subscribe to different data categories. We also describe a preliminary prototype that we have built to demonstrate this solution.

The rest of the paper is organized as follows: In Section II, we summarized related work. In Section III, we describe the system model of SEDAR. In Section IV, we describe our security and privacy design. In Section V, we describe our preliminary prototype. We conclude in Section VI by describing further work that we intend to do in the near future to complete our prototype.

II. RELATED WORK

DTN Routing Schemes

DTNs have been proposed to deal with challenging communication network scenarios e.g. networks with intermittent connectivity, large delay and extended network partitions. DTNs can be useful in many scenarios e.g. vehicular scenarios [8], wild life monitoring [10], deep space communications [2] and military networks [12]. In [4],[5], an architecture based on store-and-forward concept has been proposed. A summary of the different DTN routing approaches can be found in [14].

Secure Data Retrieval

In [15], the authors describe a privacy enhanced data-centric sensor network that allows sensor data with the same “name” to be stored in the same location so that queries for data of a particular name can be sent directly to the storing nodes using geographical routing protocols. In their design, a certain geographical area is divided into different cells. The nodes within a cell u determine the location of the storage cell v through a keyed hash function. Cell u encrypts the sensor data with its cell key and forwards the encrypted data item towards the storage cell, v . Upon receiving the message, v stores it locally. An authorized mobile sink (MS) interested in the data item can determine the storage cell v , retrieves and decrypts it with the proper cell key. The approach in [15] is only suitable for static sensor nodes. For mobile sensor networks with intermittent connectivity, the approach in [15] performs poorly especially when sensor data has short expiration time.

In [16], the authors propose a bucketing scheme to store the data items with attributes values within certain ranges, and employ encoding numbers to prevent storage nodes from maliciously dropping data. In [16], the authors assume that each publisher performs the clustering to determine the bucket size and sends the encrypted data to the nearest storage node. This means that range queries need to be flooded to all storage nodes. In addition, each publisher encrypts all the data using the same key. Our approach differs from [16]: (a) we let the storage nodes perform the clustering, (b) we select storage nodes based on the encounter opportunities, (c) we use data encryption keys that are based on subscriptions or attribute range, not on a per publisher basis. By letting the storage nodes do the clustering, the false positive (defined by the authors in [16] as the redundant data included in the response due to the fact that a publisher encrypts all data items within a certain range together) in our system will be smaller.

Content-Based Network

The authors in [17],[18] propose a content-based network where the nodes run two routing protocols: a broadcast and a content-based routing protocol. Their content-based routing protocols provide both push and pull mechanisms. The advertisements from the receivers are pushed to all potential senders. Each node that receives such advertisements will check if it already supports a content-based address (a predictor) that covers the new advertisement. If so, it simply drops the new advertisement. If not, it computes the set of next-hop links on a global broadcast tree rooted at that receiver and forwards the receiver’s advertisement along those links. The intermediate node also updates its content-based routing table. A pull mechanism is also provided in their design to allow nodes to send a send-request message to pull the

latest selected predicates from all nodes. However, the authors do not address any security design.

III. SYSTEM MODEL

Figure 1 shows the network architecture of a SEDAR system. In SEDAR, we assume that users use their mobile devices e.g. PDAs, cell-phones with Bluetooth or WLAN connectivity to communicate with servers that are deployed over a wide geographical area e.g. wireless routers within tanks. The users’ mobile devices can also communicate with one another. These devices can act as forwarding nodes for disseminated data items or queries. To achieve such forwarding, these devices run an efficient DTN routing protocol [4],[5] e.g. the enhanced Prophet scheme we design [9]. The mobile devices may also store queries and data items if the query or publish policy of the system permits. In addition, some servers that provide indexing and storage functions may be deployed. *Such servers are referred to as the indexing and storage point(ISP)s.* The deployed ISPs need not be connected together all the time since they run disruption tolerant networking protocol.

Figure 1: Network Architecture of a SEDAR system

A. Content-based Pub-Sub Model

A content-based publish-subscribe model is used in our system. Data items can be published for private or public groups. Data items that can be publicly accessed are categorized. Users can subscribe to these data categories e.g. weather or area-maps which are for public access. Data items that are published by private groups can only be accessed by users in those private groups. The private group membership can be edited by the publisher that creates such a group.

In a typical publish/subscribe system, publishers publish information in the form of data events, and subscribers have the ability to express their interests in a particular data event or group of events via subscription messages. For example, in a military content-based system, a data event that announces intelligent information for subscribers with a clearance level above 5 (assume higher value means more sensitive information) can be represented as: $e1 = \langle \text{topic, eq, intelligence} \rangle, \langle \text{clearance}, >, 5 \rangle, \langle \text{meta-descriptor, data-record} \rangle$. A subscription example could consist of the

following filter: $f1=(\langle \text{topic}, \text{eq}, \text{intelligence} \rangle, \langle \text{clearance}, \text{eq}, 3 \rangle)$ where eq denotes the keyword based matching operator and ' > ' denotes the "greater than" operator.

An access control mechanism needs to be in place such that only authorized users can access data items from certain data categories. We will discuss the security design in the next session. Here, we focus on how data items are published and retrieved.

A publisher can publish data items and store replicated copies of the data items at nodes it encounters opportunistically or at specifically deployed storage nodes called the index and storage point (ISP)s [21]. In [9], we describe how a publisher can use a friendliness metric to select more popular nodes to store its data items. Our simulation results [3],[9],[21] indicate that distributed storage of replicated data items improve the query success rate and query response time in extreme networks. Query performance results using specially deployed storage nodes called ISPs are also provided in [21]. Each storage node (including the publisher) also periodically broadcasts a data bloom filter advertising the data items that it currently stores. Such advertisement carries the sender's signature so that no node can broadcast false advertisements. A publisher can also replicate the index list of data items it has and selects more popular nodes to help advertise the index list. A publisher may decide to split the index list based on attribute values as in the Mercury approach [19]. Assume that a publisher publishes data items with two attributes. This publisher may divide the range of the first attribute into 4 bins and select 4 nodes to host indices of all data items with the values of their first attribute falling into each bin. Similarly, another 4 nodes can be selected for hosting data items with the second attribute values falling into one of the ranges.

B. Data Retrieval Schemes

Each node can subscribe to certain data categories with a mobile key server (MKS). The MKS will then distribute appropriate access keys to the node. Each node maintains a `pending_query` list to store queries that either it generates or others forward. When two nodes encounter each other, a node will send its data bloom filter (if it stores data) to the other node. Then, each node checks its pending query list to determine if the other node carries data items that match its pending queries. If there is a match, it will send those queries to retrieve the data items. Since a query sent carries the subscription filter of the original querying node, the receiving node can verify that the querying node has the authority to access the data item.

Depending on the query dissemination policy, a query may also be disseminated to any node which a querying node

encounters or to the ISPs. The query dissemination policy can also determine if one copy or multiple copies of a query are disseminated. Our previous work [3],[9] has indicated that query dissemination helps to increase query success ratio and reduces the query response time. Each node that receives a query determines (using the bloom filter it constructs for any data items it caches) if it has any data items that match the query. If there is a match, it constructs and sends a response back to the querying node. The drawback of using query duplication is that the system will have lower transmission efficiency due to the redundant replies.

IV. SECURITY AND PRIVACY DESIGN

In this subsection, we discuss our preliminary design for source privacy, and secure data retrieval.

A. Source Privacy

A publisher may decide to hide his true identity when he publishes data items. Thus, in SEDAR, a publisher owns a set of pseudonyms that he can use to generate changing identifiers for different data categories that he publishes. For example, a publisher may use an identifier constructed from the name "bob123" for data items published for the "weather" category, or the name "manmade_m55" for data items published for the "man-made" category, or the name "mingli204" for data items published for the "threat analysis" category. The identifier is the result of applying a hash function to the concatenation of a name and a counter value e.g. $\text{identifier}(\text{bob's pseudonym}, \text{weather}) = H_1(\text{"bob123"} \parallel \text{cnt}[\text{weather}])$. The counter value is incremented every time an identifier is generated. Every published data item carries the counter value that the publisher uses, and a HESS signature [20] is generated using the identifier as the publisher's identity. For example, during the week of Feb 11th, 2008, the data items Bob generates for the "weather" category may have a counter value of 3, and the HESS signature created will then be $H_1(\text{"bob123"} \parallel 3)$. Each identifier has a certain lifetime which the publisher can set. Any authorized node that retrieves a particular data item can verify the HESS signature to ensure this data item really comes from the publisher.

B. Secure Data Retrieval

For secure data management in SEDAR, we consider a data centric based solution. Access and encryption keys are used to provide confidentiality. An encryption key is used to encrypt a data item to maintain its confidentiality from the routing nodes and the nodes that have not subscribed to that data. An access key is used as a permit to access and decrypt the data. A mobile key server (MKS) is used to distribute access keys to both publishers/subscribers after they are authenticated. We use a time-based subscription

approach where all authorization permits are valid for one time epoch. A subscriber receives an access key for every data category that he is allowed to access. An authorization for a data category has a certain lifetime. The time-based subscription model allows the publisher to revoke and regenerate the access keys periodically.

Our security design is such that a user can efficiently derive the encryption keys for any data items that it has subscribed to using the access key it receives during subscription. Our hierarchical key derivation algorithm can be used for different attribute types e.g. keyword or topic based, ontology-based, string prefix/suffix or numeric based attribute. We discuss how our approach works for a category-based subscription in Section IV.B.1, and for numeric-based subscription in Section IV.B.2. Then, we discuss the advantages/disadvantages of our approach in Section IV.B.3.

1) Category-based Hierarchical Key Tree

Our approach allows different publishers to create different category-based hierarchical key trees with different root keys. Let us consider the category-based hierarchical key tree shown in Fig 2. The key tree is designed such that given a parent key, all the child keys can be easily derived but given a child key, it is computationally infeasible to derive a parent key. For example, in Fig 2, the encryption keys must be designed such that a subscriber that wants to access data items that belong to man-made category, and another subscriber that wants to access data items in the terrain category can derive the encryption keys used to encrypt all data items in the tunnels subcategory. The subscriber that only subscribes to data items in the weather category should not be able to derive encryption keys for any data items in the tunnels subcategory.

Let the access key for the root element (e.g. the US-Military topic in Fig 2) be $K_g = f_r(w)$ where f is a keyed pseudo-random function e.g. HMAC-MD5 [11], r is the secret-key of the data owner and w is the word “hobby” in Fig 2’s example. Then, we derive the key for an internal element with $K_{o||s} = H(K_o || s)$ where w is a binary string that represents the authorized category, and s is a variable bit string that represents a particular subcategory within that authorized category, and H is a one-way hash function e.g. MD5[13], and SHA-1 [6]. Note that $||$ denotes string concatenation.

We describe how an encryption key Ke for a data item, and an access key Kf for a subscription filter f are derived. Let kls be a binary string that represents a subcategory. For example the kls for the “man-made” subcategory in Fig 2 is 0010 001 (the space is created to allow the reader to see clearly how this binary string is constructed). The

encryption key for any data item within the category c is denoted as:

$$Ke = K_{kls}(c)$$

Let us assume that the root element for Publisher PI uses the key K_g for US-Military. To determine the encryption key, Ke , for the data items in the “tunnels” category, a publisher first determines the binary string, kls , that represents the “tunnels” category, which is 0010 001 0101. Then, $Ke = H(H(K_g || 0010) || 001) || 0101$.

Access Key

A subscriber can subscribe to any category within the category-based hierarchical tree. Assume that a subscriber subscribes to the terrain category, then he has the access key, K_{0010} . When he wants to access the data items belonging to the “tunnels” category, he knows that the kls for that category is 0010 001 0101 so he needs to perform the following operation to his access key to derive $K_{tunnels}$, the encryption key for data items that belong to the “tunnels” subcategory.

Fig. 2: Category-based hierarchical key tree

Note that a subscriber is allowed to subscribe to multiple categories. For example, a subscriber may subscribe to items related to both “weather”, and “man-made” categories. This subscriber will be given the access key for the weather category i.e. K_{0001} , and the access key for the man-made category i.e. $K_{0010 001}$.

The category tree design needs to be flexible enough to accommodate arbitrary updates. Prefix-based labeling schemes [22] have been proposed for document descriptions. Our access key approach can work with such prefix-based labeling too. For example ACM documents can be classified into multiple categories and each category may have different levels of sub-categories e.g. H.3 is the information storage and retrieval category, and H.3.3 is the information search and retrieval subcategory. In [22], the authors propose a DeweyID template that works more efficiently with arbitrary insertions. We replicate Figure 4 and Table 2 from [22] here to ease our discussion on how

our key-building technique can be applied to systems that support such prefix-encoding of categorization labels. Consider the categories 1.7 and 1.7.11. Using Table 2, the label 00010111 is used to encode 1.7 and the label 00010111.1000011 is used to encode 1.7.11. The authorization key for category 1.7, $K_{1.7}$ is generated as follows: $H(K_g|0111)$ where K_g is the key for category 1 with a label “0001”. The authorization key for category 1.7.11, $K_{1.7.11}$, is generated as follows: $H(K_{1.7} | 1000011)$.

A data item may be classified into different categories. To accommodate this, one can store different copies of the same data encrypted using a key from each category the item is classified into (e.g. $E(d_1)_{K_1}$, $E(d_1)_{K_2}$, $E(d_1)_{K_3}$) or create one copy of the data using a hash function of the concatenation of all keys (e.g. $E(d_1)_{H(K_1||K_2||K_3)}$). The first approach consumes more storage space if the data object size is large. If the second approach is used, then, each category key will be used to find a pointer to a memory cell which stores the concatenation of other missing keys so that the final data encryption key can be derived. Another approach is to use the attribute-based encryption approach [23].

Table 2: Prefix-free Coding Scheme [22]		
Code C_i	L_i	Value Range
0	3	1-7
100	4	8-23
101	6	24-87
1100	8	88-343
1101	12	344-4439
11100	16	4440-69,975
11101	20	69,976-1,118,551
11110	24	1,118,552-17,895,767
11111	31	17,895,768-2,165,379,414

TL	C_0	E_0	C_1	E_1		C_k	E_k
----	-------	-------	-------	-------	--	-------	-------

Fig 4: DeweyID Template [22]

2) Numeric-based hierarchical key tree

Numeric attribute based subscription is used for subscribing to a certain range of numeric attributes. For example, let us assume that Publisher P1 archives sensor data that span 60 months (5 years) starting from Jan 2002. A subscription filter in the form: $f = \langle date, \varepsilon, (l, u) \rangle$, indicates that the filter matches any sensor data collected with a date of $(date, w)$ if and only if $l \leq w \leq u$. As in Section IV.B.1, we can construct a hierarchical key tree for a numeric attribute. Assume that the value of the numeric attribute spans a certain range, then we can divide the range into different bins. Both uniform and non-uniform bin size can be used. Assume data items have an attribute with a value that ranges from $[0, 64]$. To construct non-uniform bin size, we first group data items using a uniform bin size of 2 and estimate the probability that data items fall into each bin. Then, we group the different bins together until their total probability is 0.25. In the end, we have four buckets with different bin sizes. All data items in the same bucket will be encrypted with the same key and tagged with a unique tag,

$\langle \text{tag}, [l, h] \rangle$, where tag is a binary string that represents a bin, and $[l, h]$ represents, the lower/upper bounds of the bucket.

The following example shows how we can derive encryption keys for data items that have both category-based and numeric-based attributes. For example, let us assume that we need to represent the temperature reading within a certain geographical area. We use the DeweyID-like template shown in Figure 3: TL_0 , TL_1 are the length fields for the codes that represent the region and date categories respectively. The concatenated strings $C_0||E_0$ and $C_1||E_1$ represent Region 10 and Month 12 respectively. For example, using the Table 2 in [20], the identifier for Region 10 is 100 0010, and the identifier for Week 12 will be 100 0100.

TL_0	C_0	E_0	TL_1	C_1	E_1	BL_0	B_0	BL_1	B_1
--------	-------	-------	--------	-------	-------	--------	-------	--------	-------

$C_0 || E_0 = 100\ 0010$ – Region 10

$C_1 || E_1 = 100\ 0100$ – Week 12

B_0 - t_{min}

B_1 - kid

Fig. 3: Encoding Scheme for Temperature Reading in a Particular Region

We can divide the temperature range (t_{min}, t_{max}) into m equal bin sizes. To get m equal-size bins, the bin size will be R/m where $R = (t_{max} - t_{min})$. Assume further that we round off the temperature to the nearest degree, then, given a temperature value w , we can map it to a key tree identifier $kid(w)$ which is the m -bit binary representation of the number $kid(w) = \left\lfloor \frac{w}{S} \right\rfloor$ where S is the bin size and $m = \log_2$

(R/S) . To represent the particular bin where the temperature w falls into, we need to represent t_{min} and $kid(w)$ using two binary strings, B_0 with length BL_0 and B_1 with length BL_1 . Let K_g be the root key for a particular geographical region. The access key for data that falls into Region 10 and Week 12 is $K_{access} = H(H(K_g || 100\ 0010) || 1000100)$. Further, let us assume that $R=20$, $S=5$, then $kid(15)=3$. Thus, the encryption key for the 3rd bin for Region 10, Week12 is $K_e(15) = H(K_{access} || 11)$.

3) Rekeying

In our system, one can perform a time-based or emergency-based rekeying. For time-based rekeying mechanism, the subscribers need to refresh their subscriptions at the beginning of a new period to get the new access keys. One can select different periods for different category levels e.g. Level 5 access keys may be refreshed every month while Level 4 will be refreshed only bimonthly, etc. For emergency-based rekeying, a particular subcategory key needs to be updated upon identifying a misbehaving subscriber. It is assumed that all nodes that store the data items honestly delete all data items at the end of a subscription period.

4) *Advantages of our security approach*

Scalability

Our approach allows the publisher to be stateless and ensures that the cost of handling a new subscription is independent of the number of subscribers and the locations of the subscribers a publisher currently has. In addition, each publisher needs to store only his root key unlike the traditional approach where he needs possibly 2^N keys with N subscribers. The number of keys maintained by each subscriber depends on the number of categories that he/she subscribes to. Thus, our approach is more scalable.

Communication Cost

With the traditional subscriber group approach, changes to the group keys are needed whenever a new subscriber joins. This means the number of keys stored by the publisher will change with different numbers of subscribers, and that the existing subscriber needs to be informed of new keys which incur communication cost. However, with our approach, there is no need to update any key as long as new subscribers are allowed to access pre-existing data items in the categories they wish to subscribe.

5) *Disadvantage of our security approach*

When a data item can be classified into multiple categories, our approach requires us either to replicate stored copies of the data item or store encrypted copies of missing keys if we use a concatenation of all category keys as the data encryption key. Our approach lacks the flexibility offered in the attribute-based encryption approach [23]. In ABE, access control policy based on conjunctives and disjunctives of different attributes can be defined. We intend to explore a hybrid solution based on both approaches in the future.

V. PRELIMINARY PROTOTYPE

We have built a preliminary prototype. Our prototype is written in C++ and runs on laptops running Ubuntu. There are several important modules in our prototype: namely a publisher, a client, a storage, and an authentication and key distribution module. A four-node testbed is built. One node, *rcm2*, acts as the mobile key server (MKS) with whom any publisher or subscriber can authenticate themselves and obtain the authorized access keys for the data categories they are authorized to access or publish. A second node, *firefox*, acts as a storage node which stores all the data items that publishers send. A third node, *monkeywrench*, and a fourth node, *phoenix*, each runs both a publisher and a subscriber module. A node needs to authenticate itself using a password with the mobile key server. After it is authenticated, it will get a file containing access keys for all the authorized data categories. Figure 4(a) shows the outputs at the MKS where the 2 nodes obtain their access key files. Figure 4(b) shows the data items retrieved by the subscriber running on the node *phoenix*. Figure 4(c) shows the data items stored at the storage node. These data items are sent to the storage node by the two publishers running

on the nodes, *monkeywrench* and *phoenix*. Figure 5 shows the encryption/decryption time for different data item sizes. The experiment was conducted on a laptop running our data centric security solution. The laptop runs Ubuntu and has a 2.0GHz Intel Pentium M processor with 1GBytes RAM, 2 Mbytes CacheSize. The results indicate that the encryption and decryption times using our approach is reasonable.

VI. CONCLUSION

In this paper, we have described a mobile information retrieval system that we are building. Specifically, we describe our system model, publish/subscribe model, data retrieval schemes and a data-centric security approach that we have designed. Our data-centric security design allows publishers/subscribers to derive encryption keys for all subcategories that are within their authorized data categories. Our solution is more scalable than the traditional subscriber groups-based security solution. We have also built a preliminary prototype. Additional work needs to be done before the system can be deployed. For example, we need to design a security solution to mitigate against selective dropping attacks by any compromised storage nodes. As for our prototype, we need to integrate our prototype with an ontology framework. Then, we need to do a full-scale integration test using a bigger scale testbed with more nodes.

ACKNOWLEDGMENT

This work is sponsored by Defense Advanced Research Projects Agency (DARPA) under contract W15P7T-06-C-P430. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. This document is approved for public release, unlimited distribution.

REFERENCES

- [1] S. Das, C. Perkins, E. Royer, "Performance Comparison of Two On-demand Routing Protocols", Proceedings of IEEE Infocom, March 2000.
- [2] S. Burleigh et al, "Delay Tolerant Networking – an approach to interplanetary internet", IEEE Communication Magazine, June, 2003
- [3] M. Chuah, P. Yang, "Performance evaluation of data-centric information retrieval schemes for DTNs", Proceedings of Milcom, Oct, 2007.
- [4] K. Fall, "A delay-tolerant network architecture for challenged Internets", *Proceedings of SIGCOMM'03*, August 2003.
- [5] V. Cerf et al, "Delay-Tolerant Network Architecture", Internet Draft, draft-irtf-dtnrg-arch-02.txt, July 2004.
- [6] D. Eastlake, P. Jones, "US Secure Hash Algorithm 1(SHA1)", RFC3174, Sept 2001.
- [7] A. Lindgren, A. Doria, O. Schelen, "Probabilistic routing in intermittently connected networks", *Sigmobile, Mobile Computing and Communications Review*, Vol 7(3), pp 19-20, 2003.
- [8] J. Burgess, B. Gallagher, D. Jensen, B.N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking", to appear in Proceedings of IEEE Infocom, April, 2006.
- [9] M. Chuah, P. Yang, "Data-Centric Information Retrieval Schemes for DTNs", CSE Technical Report, March, 2008.
- [10] P. Juang, etc, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet", *Proc ASPLOS*, Oct 2002.

- [11] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", Feb, 1997.
- [12] DARPA DisruptionTolerant Networking Program, <http://www.darpa.mil/ato/solicit/DTN>, July, 2006
- [13] R. Rivest, "The MD5 Message Digest Algorithm", RFC1321, April, 1992
- [14] Z. Zhang, Q. Zhang, "Delay/Disruption Tolerant Mobile Ad Hoc Networks: Latest Developments", WCMC, Vol 7, Issue 10, pp 1219-1232.
- [15] M. Shao et al, "pDCS: security and privacy support for data-centric sensor networks", Proceedings of Infocom, 2007.
- [16] B. Sheng, Q. Li, "Verifiable Privacy-Preserving Range Query in Two-tiered Sensor Networks", to appear at IEEE Infocom, 2008.
- [17] A. Carzaniga et al, "A routing scheme for content-based networking", Proceedings of Infocom 2004.
- [18] A. Carzaniga and A. L. Wolf, "Content-based networking: a new communication infrastructure", NSF Workshop on an Infrastructure for Mobile and Wireless Systems, Scottsdale, Arizona, Oct 2001.
- [19] A. R. Bharambe et al, "Mercury: supporting scalable multi-attribute range queries", Proceedings of ACM Sigcomm, Aug/Sept, 2004.
- [20] F. Hess, "Efficient Identity based Signature Schemes based on Pairings", Proceedings of SAC 2002, LNCS 2595 p310-324, St Johns, Newfoundland, Springer-Verlag, 2003.
- [21] M. Chuah, J. Han, "Performance studies of information retrieval schemes for multi-attribute queries in DTNs", CSE Technical Report, March, 2008.
- [22] T. Harder et al, "Node labeling schemes for dynamic XML documents reconsidered", Data & Knowledge Engineering, Vol 60, Issue 1, pp126-149, Jan, 2007.
- [23] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-Policy Attribute-Based Encryption", Proceedings of IEEE Symposium on Security and Privacy, pp 321-334, 2007.

(a) output from the mobile key server

(b) output from the subscribing node

(c) Output from the storage node

Figure 4: Screenshot from different nodes

Figure 5: Encryption/Decryption Time for Different Data Item Sizes