Evaluation of Path Recording Techniques in Secure MANET

Danai Chasaki and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
Email: {dchasaki,wolf}@ecs.umass.edu

***Abstract*—Secure operation of mobile ad-hoc networks requires the ability to track the flow of traffic through the network. Knowledge of the path of a packet allows inference of correct operation or potential attacks. In prior work, we have described a number of different techniques for recording such path information and ensuring its correctness through cryptographic techniques. In this paper, we evaluate these path recording techniques in a realistic military scenario. Using topology information and traffic traces from the Lakehurst scenario, we provide a quantitative evaluation of our techniques and determine the overhead for path recording in terms of data structure size and computational requirements.**

I. Introduction

Mobile Ad-Hoc Networks (MANET) are an essential component of the military communication infrastructure. Secure communication within such networks and the ability to defend against potential attacks are essential characteristics of military MANET [1], [2]. Recent research has aimed at designing MANET with intrinsic information assurance capabilities [3]. The inherent design of such networks should not allow misbehaving nodes to injecting traffic or launch denial of service attacks.

Solutions to providing security in mobile ad-hoc networks are manifold and need to operate in concert to successfully defend against attacks: capabilities-based networks implement an off-by-default behavior that requires traffic to positively authenticate itself on all hops in the network [4]; network-coding and its underlying path diversity can circumvent malicious nodes that do not forward traffic correctly [5]; packet paths can be recorded to validate control path information against data path operations [6]. In conjunction, these and other techniques can help in providing a network infrastructure that by design is less vulnerable than existing TCP/IP-based networks.

This paper focuses on path recording, which is mechanism to track the flow of traffic through the network. Typically, there are two sources of information from which topology information can be obtained:

- Topology information via control plane: Routing message exchanges between nodes (e.g., link state updates) provides a view of the current state of the network.
- Topology information via data plane: Data transmission along paths in the MANET can only travel along valid links and thus implicitly reflect topology information.

Our path recording techniques use topology information obtained by the data plane. For every packet that traverses a path in the network graph, the IDs of the nodes that it passes through is recorded in the packet header. When the packet reaches the end-node, the receiver obtains the complete path record. At this point the receiver can compare the path record to the routing information he has received from the other nodes and check for inconsistencies. A mismatch indicates malicious behavior on the sender's side. We discussed algorithms and data structures for path recording in our prior work [6].

In this paper we evaluate some of these path recording techniques on a realistic operational scenario. We evaluate the performance of these path recording methods in terms of overhead (i.e., space requirements in packet header) on the "Lakehurst" scenario, which is commonly used in MANET evaluations [7]. The specific contributions of this work are:

- An evaluation of the performance of path recording techniques on a realistic network scenario.
- A comparison of the packet overhead for path recording against the theoretical optimum for a given scenario.

The remainder of the paper is organized as follows. Section II introduces related work. The specific path record data structures discussed in our prior work are briefly described in Section III. Their evaluation on the Lakehurst scenario is presented in Section IV. Section V summarizes and concludes this paper.

II. Related Work

There has been work on obtaining topology information from a network in different domains. One of the most efficient and widely used methods to extract information about the nodes present in a network and the order in which these nodes appear is network tomography. Topology reconstruction techniques based on end-to-end delays of multicast traffic are proposed in [8] to infer the multicast tree. Tian and Shen also propose an algorithm which determines the topology of a network based on end-to-end measurements in [9]. Probe packets are sent from some sources towards multiple destinations, and each pair of nodes keeps track of the packets received. The nodes on which multiple links converge share

This material is based upon work under a subcontract #069153 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-08-C-2013.

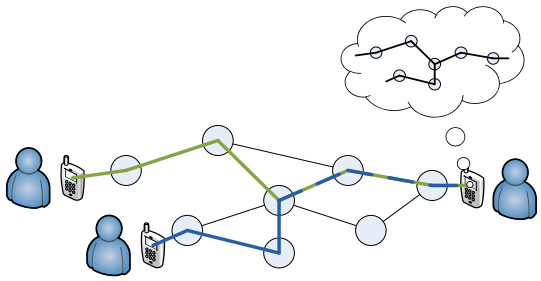


Fig. 1. Path recording and reconstruction.

the information about packet loss or delay of multiple links. Thus, the correlation of the received information can be compared and through statistical methods the whole network tree can be reconstructed.

Savage in [10] has proposed a scheme of recording topology information inside the packets. While a packet traverses the network adjacent nodes insert randomly adjacent edge information into its ID field. So after a large number of collected packets the topology can be reconstructed. A similar method is an algebraic approach where information is inserted into the packets randomly in the form of polynomials [11].

Mobile agents have also been used as a solution to the topology discovery problem. A mobile agent is a controllable program that can move inside a network. While traversing the network, it is capable of performing certain operations on each individual node it visits, cooperating with other mobile agents as well. Mobile agent deployment is proven to be very efficient and scalable, thus extremely useful in large networks, e.g. wireless networks. In the context of topology discovery, many mobile agents are generated in the network to collect topology information; they travel from node to node periodically gathering information and transmit it back to a centralized management station [12].

Our approach to topology reconstruction differs to these inference approaches insofar that we explicitly record the path that a packet takes through the network. This requires a change in the packet header and packet forwarding routine. In the context of secure MANET, this is a reasonable assumption since their security design requires many other additional changes.

III. Path Recording Techniques

Path recording aims are tracking the path of a packet through the network. This process is illustrated in Figure 1. Traffic carries the recorded path information; the receiver verifies the record and reconstructs the path. This information, which reflects the network topology in the data plane, can then be compared to topology information exchanged in the control plane. Discrepancies indicate potentially malicious behavior by participating nodes.

In [6] we described five different algorithms and their data structures that can be used as the path record: node append, bit vector, prime number IDs, sampling, and Bloom filter. The first three are deterministic methods while the last two are of a probabilistic nature. Our analysis quantifies the packet overhead (packet header size required to store the data structure) and the reconstruction time (on the receiver side). These metrics are important since they determine the bandwidth overhead in the network as well as the processing overhead on the systems that verify the path.

TABLE I
Path record for different types of paths

Path record	unicast	multicast/ network coding
unordered set of nodes	no	no
ordered sequence of nodes	yes	no
unordered set of edges	yes	yes

Before discussing the performance of these techniques in a realistic environment, we provide a brief overview of the path recording techniques we described in [6].

A. Requirements of Path Recording

Depending on the use of path recording, it may be necessary to record the *nodes* of the network that were traversed or the *edges* (i.e., links) that were traversed. The type of the network on which we are performing path recording and the type of information we store in the path record play a big role in our ability to reconstruct the path. For unicast paths it is sufficient to record either the edges a given packet is passing through or the nodes in the order they are traversed. For more complicated paths that involve network coding (see [5]), multiple paths may need to be recorded to reflect how the packet mixture was created. Thus, the topology can be reconstructed on the receiver side only if it receives the specific edges that have been traversed. Table I summarizes for which type of record a reconstruction of unicast or multicast/network coding is possible.

B. Path Recording Data Structures

In prior work, we discussed how to record the path that a packet takes and how to reconstruct this information at the receiving end. We explored different data structures that can be used store the path record, which is essential for the path reconstruction.

There are two fundamentally different approaches to recording path information: deterministic path recording and probabilistic path recording. Deterministic approaches reconstruct the path by keeping all the path information in a packet. The entire path can be reconstructed without error from a single packet. Probabilistic approaches record partial path information in a packet. Reconstruction may be inaccurate or require multiple packets.

In this paper we focus on two deterministic and one probabilistic method, which – based on our theoretical results – have a good trade off as far as the packet header size, and the reconstruction effort are concerned. These methods are illustrated in Figures 2 and 3.

- **Node Append:** In this straightforward method, each node on the path appends its ID to a variable length header

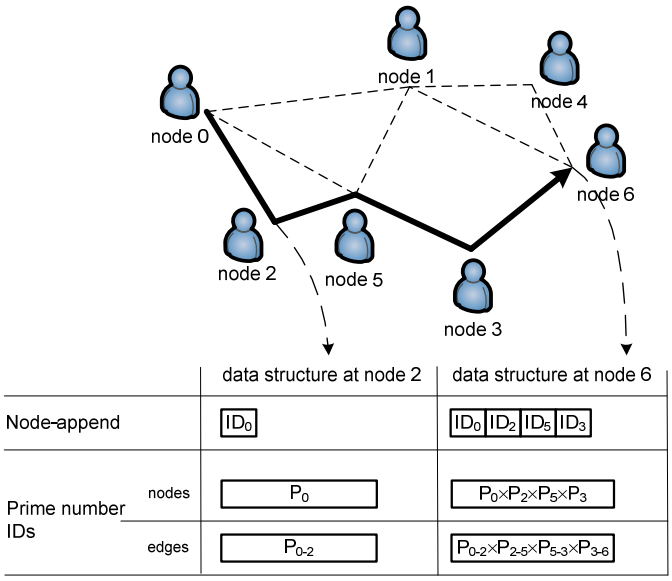


Fig. 2. Deterministic path recording methods.

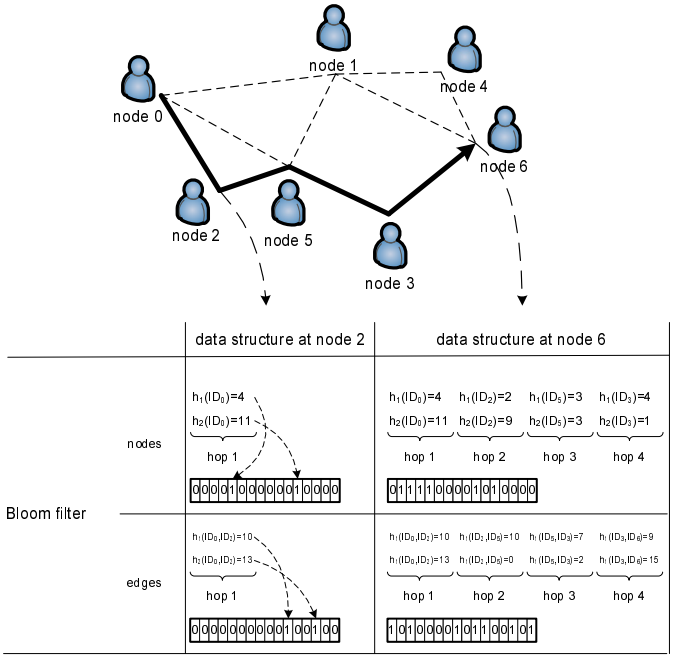


Fig. 3. Probabilistic path recording method.

field of the packet. The receiver extracts the sequences of node IDs to obtain the ordered list of traversed nodes and edges.

In the example shown in Figure 2, the packet starts traversing the path at node 0, and arrives at node 6 passing through node 2, 5, and 3. When node 2 receives the packet, the value in the header field is ID_0. Node 2 appends its ID (i.e., ID_2) next to ID_0. The same procedure is followed in subsequent nodes.

This method can only record a single path, not a subgraph that represents a network coding mixture since edges are inferred from the linear sequence of nodes.

- **Prime Number IDs:** In this method, each node or edge is assigned a prime number as an ID. The path record carries the product of all visited nodes/edges. The received integer is factorized to obtain the IDs of nodes/edges involved.

  An example of the process is illustrated in Figure 2. This method can record an arbitrary subgraph and thus is suitable for network coding.
- **Bloom Filter:** A Bloom filter is a data structure that can efficiently store membership information [13]. To add an element to a Bloom filter, several hashes of the element are computed. The bits at the bit positions provided by the hash functions are set to 1. To test for elements in the Bloom filter, the hashes are computed and it is checked if the corresponding bits are set. When testing for membership information, it is possible to obtain false positive answers.

  A key parameter for the Bloom filter is its size. Larger Bloom filters yield lower false positive rates and thus decrease the number of required aggregate signature tests (for aggregate signatures see [14]). The Bloom filter that records edges is suitable for network coding.

For additional details on these methods and the formal analysis of their performance, see [6].

IV. Evaluation of Path Recording in Lakehurst Scenario

We now turn to the question of which of these path recording techniques is more efficient. The main concerns in terms of efficiency are the following three quantitative performance metrics:

- Data structure size: The amount of space needed for storing the path record data structure determines the size of the header field necessary for recording.
- Recording time: The amount of processing time it takes to update the path record data structure in every node that is being traversed.
- Reconstruction time: The computational time required for reconstructing the packet path from the path record determines the computational overhead on the receiver.

Since prior analysis indicates that the data structure size is the main concern in path recording we use the first metric as our basic comparison point of the path recording techniques.

A. Lakehurst Trace Analysis

For the performance evaluation of the several path recording methods we have used real time data from the Lakehurst scenario [7]. The network topology for this particular trace consists of 30 nodes and the connection requests are at most 6 hops apart. The nodes represent vehicles, most of them located on the ground (Jeeps) and a few in the air (UAVs, strike fighter, medevac). We analyzed the performance of the network for a trace of 185 minutes. The network graph is

being refreshed every 5 minutes, yielding 28 intervals over the time. The connectivity of a pair of nodes in an interval is determined by the pathloss (dB) of the link that connects the two nodes. We explore pathloss cutoff scenarios of 40 % (i.e., low connectivity) and 80% (i.e., high connectivity). The ground links and the air to air links are well connected. The air-to-ground links are the bottleneck of the network connectivity.

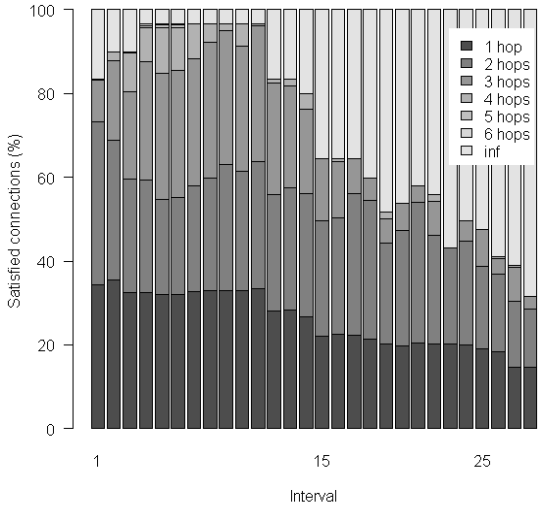
First of all, we identify the path size between two potential nodes that wish to communicate. The path size in other words is the number of hops that a packet needs to travel, when two specific nodes want to communicate with each other. We look at all the possible edges/pair of connected nodes and record how many hops need to be traversed for every connectivity attempt. Our simulations show that for the whole duration of the Lakehurst trace (for all intervals) no more than 6 hops need to be traversed for two nodes to connect to each other. In Figure 4, we show histograms of how many connections need how many hops, over all recorded intervals. No connectivity between two nodes is indicated by "inf". The histograms show that as we move towards the last scenarios/intervals of the Lakehurst trace the percentage of connections that cannot be established is increasing. Moreover, we can see that the average path size is 2–3 hops.

B. Path Recording Overhead

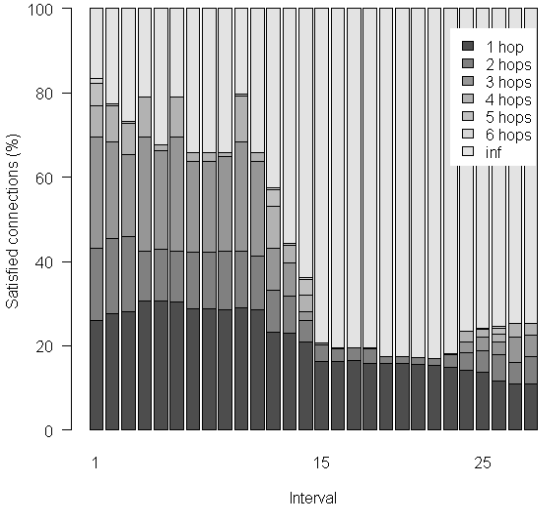
We evaluate two of the deterministic methods (node append and prime numbers) and compare their performance to a theoretical optimum. The theoretical optimum is the lower bound on the space requirement of a path recording method. While we may not know a practical method that obtains the lower bound, we know that such a function would need to fulfill the following requirement: Every possible combination of node/edge IDs should map to a distinct element in the image set and the size of image set should be minimal (i.e. injective if not bijective).

Figures 5, 6, and 7 show node append, prime numbers, and the optimum, respectively. Each figure shows the percentage of connections (y-axis) that can be satisfied with the number of bits allocated shown on the x-axis. Ideally, we would like 100% of the connections to be satisfied. The case of non-satisfied connections is the case where the number of bits available in the packet header do not suffice to correctly record every node that the packet traverses from source to destination. In this case a non-complete path is recorded. Since we want to make sure that the receiver reconstructs the correct paths, we can make sure that incomplete path records do not reach the destination node. When a node tries to append his ID to the packet header but there are no more bits available, then the so far recorded path is removed from the header and the recording operation is reset

Each black line shows the performance for one of the intervals in the Lakehurst scenario. It is clear that as we move towards the last intervals less and less bits are required to approach 100% satisfied connections. The first intervals seem to be denser and more expensive in terms of size requirements.



(a) 40% connectivity threshold.



(b) 80% connectivity threshold.

Fig. 4. Histogram of how many connections need certain number of hops over all intervals in cases of 40% and 80% connectivity threshold.

The colored line, which consists of the minima for each x-value, represents the worst case performance (i.e., least number of connections satisfied).

A direct comparison between the worst-case performances of all three approaches is shown in Figure 8. Each line is taken from the evaluation results above. The results show that node append performs closely to the optimum and requires only about 10 bits more in space when all connections need to be satisfied. Prime numbers performs less well and requires nearly twice as many bits as the optimum. However, it may still be desirable to use this approach if a variable-length list (as used in node append) is not desirable.

Turning now to the evaluation of the Bloom filter method, we should keep in mind that due to its probabilistic nature false positives will always occur in path recording. Since larger Bloom filters yield lower false positive rates, we are using a large enough Bloom filter of size 512 bits for our simulations.

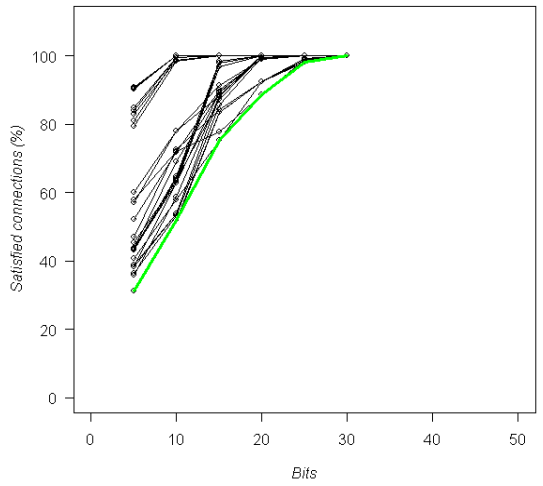


Fig. 5. Node Append: number of satisfied connections vs. size for all 28 scenarios.

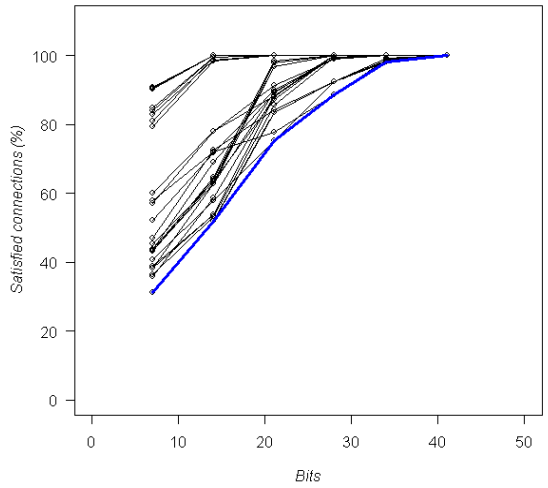


Fig. 6. Number IDs: number of satisfied connections vs. size for all 28 scenarios.

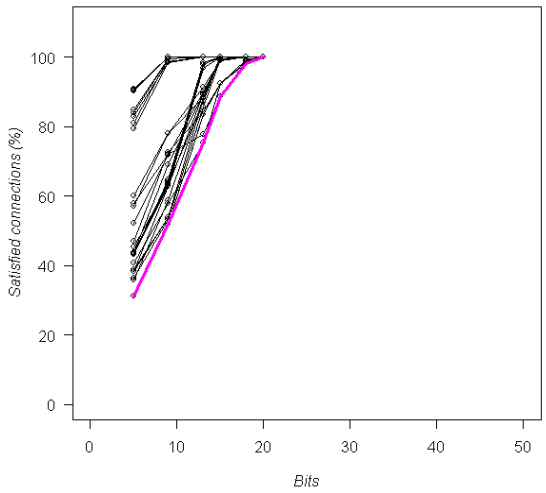


Fig. 7. Optimum: number of satisfied connections vs. size for all 28 scenarios.

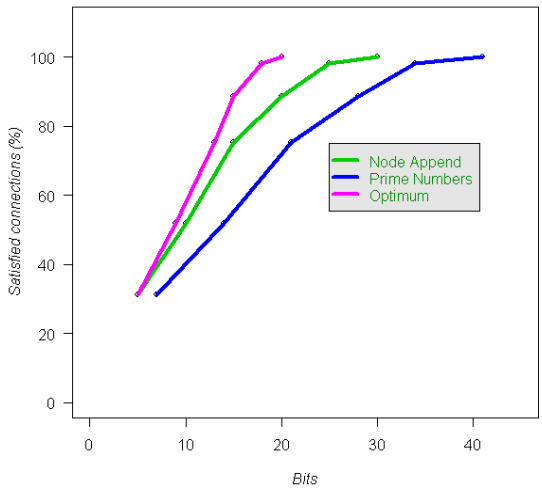


Fig. 8. Comparison of Node-Append, Prime Number IDs and Optimum.

TABLE II
FALSE POSITIVE EDGES IN BLOOM FILTER.

	Bloom filter without connectivity check	Bloom filter with connectivity check
0 extra edges recorded	8.4%	13.9%
1 extra edges recorded	89.4%	84.5%
2 extra edges recorded	2.3%	1.6%
average number of signature checks	2.90	2.77

The length of the data structure is fixed for this method, and obviously it is much larger than any of the data structures used in the previously evaluated deterministic methods. However, this still might be a very useful method for different network configurations, i.e. sparser network topology.

We use the number of errors that occur when the Bloom filter method is used, as the performance metric of this path recording method. We determine the number of false positive edges that we get in the Lakehurst scenario. The definition and formulas to derive the false positives of a Bloom filter are described in detail in [15].

We first construct the Bloom Filter for each one of the possible connections, over all 28 scenarios in the Lakehurst trace. The procedure that we follow for each of the paths is as follows:

1) We add an element-path to a Bloom filter by computing several hashes of the element. The bits at the bit positions provided by the hash functions are set to 1. In this way a sequence of k edges is recorded in the Bloom Filter.
2) We test all possible edges to see if they belong to the Bloom filter by computing the hashes and checking if the corresponding bits are set. Out of these tests a sequence of n edges comes out.
   - If n=k then we do not have any false positives.
   - If n>k this means that there are false positives due to the small size of the Bloom Filter.

   We record the difference (n−k), which gives us the

number of extra recorded edges (false positives).

Each false positive edge leads to problems when reconstructing the path. The false information can be detected by the aggregate signature check, but it requires a trial-and-error approach to determine which edge(s) have been falsely added. The number of checks against the aggregate signature is $\binom{n}{k}$.

We should mention here that it is very expensive to perform tests against the aggregate signatures. The most desired scenario would be the one where we would have to do minimal testing (0 or 1 checks).

Since our goal is to reduce the number of signature checks, we should reduce the number of extra recorded edges. For that, we have implemented a "connectivity check" that removes edges that are not part of a connected graph (indicating that they are due to false positives). Table II shows the results from the Lakehurst scenario. Only 8.4% of connections have no false positives. Using the connectivity check, this number is increased to 13.9%. This improvement is modest, which can be explained by the high connectivity within the Lakehurst network topology. False positive edges are likely connected to a part of the graph and thus cannot be identified in the connectivity check. We expect that the performance would be better in a more widely spread, less dense network.

In summary, the Bloom filter approach generates too many false positives in the Lakehurst scenario and thus is not practically useful. However, the deterministic approaches to path credentialing work well. As expected from our analysis, node append performs close to the theoretical optimal and is the overall recommended solution.

V. Summary

The record of the path of a packet through a MANET can be used to infer correct operation of the network. Such a mechanism can also defend against potential malicious behavior by some users in the network that attempt to advertise incorrect connectivity information. We evaluate the performance of different methods for recording the identifiers of nodes and links in the network and eventually obtaining the complete network graph. Our evaluation provides quantitative tradeoffs between the described techniques and determines the overhead for path recording in a realistic military scenario.

References

[1] *Network Centric Warfare*, Department of Defense, Washington, DC, Jul. 2001, report to Congress.

[2] *Global Information Grid Architectural Vision*, Department of Defense, Washington, DC, Jun. 2007.

[3] S. Alexander, B. DeCleene, J. Rogers, and P. Sholander, "Requirements and architectures for intrinsically assurable mobile ad hoc networks," in *Proc. of the 2008 IEEE Conference on Military Communications (MILCOM)*, San Diego, CA, Nov. 2008.

[4] T. Wolf, "A credential-based data path architecture for assurable global networking," in *Proc. of the 2007 IEEE Conference on Military Communications (MILCOM)*, Orlando, FL, Oct. 2007.

[5] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[6] D. Chasaki, Y. S. Hanay, and T. Wolf, "Topology reconstruction via path recording in secure MANET," in *Proc. of the 2008 IEEE Conference on Military Communications (MILCOM)*, San Diego, CA, Nov. 2008.

[7] D. Caprioni and A. Russo, "Small unit operations situation awareness system (SUO-SAS) radio architecture and system field testing results," in *Proc. of the 2003 IEEE Conference on Military Communications (MILCOM)*, Monterey, CA, Oct. 2003, pp. 198–203.

[8] N. G. Duffield and F. Lo Presti, "Network tomography from measured end-to-end delay covariance," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 978–992, Dec. 2004.

[9] H. Tian and H. Shen, "Multicast-based inference of network-internal loss performance," in *Proc. of 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 2004)*, Hong Kong, China, May 2004, pp. 288–293.

[10] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for IP traceback," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 226–237, Jun. 2001.

[11] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *ACM Transactions on Information System Security*, vol. 5, no. 2, pp. 119–137, May 2002.

[12] H. Tian and H. Shen, "Mobile agents based topology discovery algorithms and modelling," in *Proc. of 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 2004)*, Hong Kong, China, May 2004, pp. 502–507.

[13] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[14] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. of International Conference on the Theory and Applications of Cryptographic (EUROCRYPT 2003)*, ser. Lecture Notes in Computer Science, vol. 2656. Warsaw, Poland: Springer Verlag, May 2003, pp. 416–432.

[15] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," in *Proc. of the 40th Annual Allerton Conference on Communication, Control, and Computing*, Allerton, IL, Oct. 2002, pp. 636–646.