Collaborative Location Certification for Ad-Hoc Sensor Networks

Sol Lederer Jie Gao Radu Sion
Computer Science, Stony Brook University
{lederer, jgao, sion}@cs.stonybrook.edu.

Abstract— **Location information is of essential importance in ad-hoc sensor networks deployed for generating location-specific event reports. When such networks operate in hostile environments, it becomes imperative to guarantee the correctness of event location claims. In this paper we address the problem of assessing location claims of un-trusted (potentially compromised) nodes. The mechanisms introduced here prevent a compromised node from generating illicit event reports for locations other than its own. To achieve this goal, in a process we call *location certification*, data routed through the network is "tagged" by participating nodes with "belief" ratings, collaboratively assessing the probability that the claimed source location is indeed correct. The effectiveness of our solution relies on the joint knowledge of participating nodes to assess the truthfulness of claimed locations. By collaboratively generating and propagating a set of "belief" ratings with transmitted data and event reports, the network allows authorized parties (e.g. final data sinks) to evaluate a metric of trust for the claimed location of such reports. Belief ratings are derived from a data model of observed past routing activity. The solution is shown to feature a strong ability to detect false location claims and compromised nodes. For example, incorrect claims as small as 2 hops (from the actual location) are detected with over 90% accuracy.**

I. Introduction

Node location information plays a fundamental role in ad-hoc networks. Specifically, in sensor networks, it is critical that the reported location of all nodes are accurate, to ensure a point of reference for location-specific applications. Thus, a robust network must require such information be uncompromised, lest a few faulty or malicious nodes will have a deleterious effect on the entire network.

Existing work investigates secure localization [11], i.e., how nodes determine their own location in a hostile environment, and secure location verification [15, 16], determining the location of a node in the face of liars. Typically these protocols involve special anchor nodes, or nodes whose location is not corruptible. Based on the distance to these nodes, the location of the remaining nodes is determined with certain assurances by deploying distance-measuring RF or ultrasound-based mechanisms and performing multi-way handshakes under synchronized clocks assumptions. These methods are designed to be used when the network is first deployed, to establish the location of all nodes during its initial setup.

However, when operating in hostile environments, it is essential to secure location information claims *at runtime*, in the presence of compromised nodes, that could falsify location claims and inject incorrect event reports into the information stream. False location information may lead the data sink to take action in a location where none is warranted, and vice versa, not take action in the area where a response is necessary.

Radu Sion is partly supported by the NSF Cybertrust award CNS-0627544 and by the Stony Brook Office of the Vice President for Research.

We introduce a protocol that validates the truthfulness of location information associated with event reports. The protocol relies on the collaborative interaction of the network nodes to find compromised parties. Since each node is an active participant in the network and spends a substantial amount of time and resources relaying messages for others, it automatically has some knowledge of the activity within the network. This knowledge can be put to good use in spotting anomalous behavior. The workload and detection ability is thus distributed across the network, to avoid a single point of failure and gracefully degrade with increasing number of compromised nodes.

To achieve this goal, at an overview level, nodes in the network (compactly) record summaries of routing paths taken by packets through the network. Upon receiving a packet, nodes examine whether their route matches a historically expected behavior by packets from the same claimed location. A belief about the correctness of this location claim of this packet is then created and propagated to the data sink, either as part of this packet or later on, in an out of band fashion. The attached beliefs will be used by the authorized packet evaluators PEs (sinks or authorized intermediate relay nodes) to certify the truthfulness of the packets' location information.

The memory limitations of sensor nodes require light-weight protocols both in terms of memory and power usage. Accordingly, we developed a path metric and a compact way to express path trajectories, by using locality-sensitive hashing [7]. This metric captures the fact that packets from sources with incorrectly claimed locations are likely to have path trajectories deviating from previously observed traffic paths.

II. Related Work

Secure localization has been studied by a number of groups. For example, the SeRLoc protocol [11] tackles secure localization by using specially equipped "locator" nodes that emit powerful beacon signals through the networks. Depending on the beacons a node hears, it computes the "center of gravity" of the overlapping regions to determine its location. The locator devices are assumed to be tamper proof. Elsewhere [12] robust statistics are used to improve the resilience of an anchor-

based localization algorithm in a hostile environment where the nodes may receive false information from the neighbors.

Even though the nodes can obtain their locations correctly by these secure localization protocols or extra secure location support such as GPS, they do not prevent a node from lying about its location and generating event reports with a false location claim. A few existing protocols have tackled this problem for inter-node settings, mainly using fine grained timing analysis of signal travel times, under the assumption of high accuracy clocks available on sensors.

III. Model

In this section we discuss the considered adversarial and deployment models.

A. Adversary

Of concern here is a malicious, powerful adversary with strong incentives to capture and compromise sensors for the purpose of altering the sensor data flow, e.g., by inserting false data and event reports and eventually influencing decision making process at the base station. For such an adversary, pure denial of service (DOS) attacks that aim to disable sensors and parts of the network are only of marginal interest and will not be considered here. For DOS attacks, [3, 17] offer techniques to address these issues. Multi-path forwarding [9] alleviates the problem of malicious relay nodes dropping legitimate reports. Also, by using a cache to store the signatures of recently forwarded reports we can prevent against the same packet from being replayed [8, 18].

Through the process of what we call *location certification*, data routed through the network will be "tagged" by participating nodes with "belief" ratings, collaboratively assessing the probability that the claimed source location is indeed correct. We call this process

To circumvent location certification (e.g., for the purpose of injecting fake event reports referencing remote, out of reach locations) an adversary could attempt to: (i) favorably modify certificates for its own fake data (e.g., by altering the associated belief ratings), or (ii) unfavorably alter certificates of legitimate traffic. The probability of success of such attempts is naturally related to the density of compromised nodes in the network. The ability of success adapts gracefully to the density of compromised nodes and the solution can operate even in the presence of a large number of adversarial nodes, as validated through simulations.

For illustration purposes, we first consider an adversary that only attempts to maliciously claim a different location in its event reports (but does not maliciously alter belief ratings of other packets it routes). We then discuss additional security issues in section V.

B. Deployment. Routing.

We focus in this paper primarily on monitoring networks in which the sensors collect information of interest and send data/event reports to a base station (data sink) for post-processing and analysis. Immediately after deployment, for an initial short period, the network is assumed free from any adversarial presence. Since our location certification procedure is based on using past history of network routes, we must assume that the original history is initially "clean". The better the history data is in terms of "cleanliness", the better the location certification will perform.

IV. Collaborative Location Certification

In this section we detail the main components of the location certification protocol.

A. Solution Overview

At an overview level, the proposed solution unfolds as follows. Immediately after deployment, network secure localization protocols [11] allow sensors to acquire location information that is to be later used in event reports. Existing research achieves this by assuming a largely un-compromised network for a short amount of time after deployment. We believe this is reasonable, especially if we consider the minimal time and resource requirements for corrupting even an individual sensor. In other words, even in the presence of an adversary with immediate physical access to a sensor node, some amount of time (e.g. minutes) will be required to locate and compromise the sensor internals and software.

Once the network becomes fully operational, sensors will start generating event reports associated with their respective location. A compromised sensor could then attempt to generate illicit event reports for locations other than its own. To defeat such an adversary, nodes along the path from the source to destination will attach "belief" ratings to passing data packets, quantifying the correctness probability of the claimed source locations. Informally, beliefs are a function of observed past traffic patterns in conjunction with the claimed source location. Upon receiving packets with routing information deviating from expected traffic patterns, nodes will have the opportunity to propagate negative belief ratings associated with these packets. The negative beliefs reflect the appearance of an anomaly in the routing pattern.

Thus this scheme is able to detect both the case in which the routing pattern is altered by an adversary (a compromised node lies about its location, or other routing attacks such as wormhole attacks [6, 13, 14]) and the case of node failures — a large fraction of nodes run out of battery power or get physically destroyed by adversaries causing significant routing pattern changes. A node which rarely participated in the data collection operation will get a low confidence, which is reasonable as the network collectively has little or no information to decide whether it is a legitimate node.

We re-iterate that this solution assumes *the network is initially free from adversaries* for a short period of time. If a large number of compromised nodes is present at the start and they are able to generate arbitrary traffic patterns then collaborative certification will be less effective. In a typical deployment there is often a short period of time which is more than enough for our scheme to collect enough history traffic data. With this limitation in mind, we believe that the novelty in our scheme lies in the compact and efficient way of summarizing the history traffic pattern and the ability of using the history to verify the correctness of future packets.

B. Strawman's Book-keeping.

Before proceeding, to illustrate, we first discuss an extremely simple book-keeping mechanism, the understanding of which will motivate our final solution. As part of the routing protocol, each sensor will maintain a history and normalized count of each previously seen source-destination pair for routed packets. New incoming packets from rarely seen sources will then be considered more suspicious and associated with a low rating.

While this scheme is extremely simple and scalable, it presents certain limitations, in particular in its ability to detect deviations in full routes (as opposed to endpoints). If a node does route information between the claimed origin and destination, then the packet from an adversary claiming to be from a different location will be considered fair game. To achieve a better detection accuracy, more information about information flow is required in the belief rating construction.

C. Inter-Path Distance Metric

Accordingly, we explore first how to compare packet routes efficiently in a meaningful way. Based on the sequence of nodes a packet has visited, we derive a trajectory of a packet by the piece-wise linear curve connecting the intermediate nodes in their visited order.

We define a distance metric that measures how far two trajectories are. The distance is designed such that fake claimed locations for packets will result in large distances between real and expected trajectories. There are many generic ways to measure the distance between two curves in space, such as Hausdorff distance and Frechét distance. Here we design a measure well suited to our problem at hand.

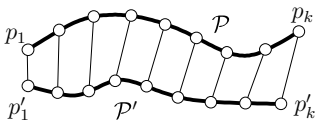

Fig. 1. The distance metric between two paths $\mathcal{P}, \mathcal{P}'$. In this figure we adopt a uniform parametrization and the samples are placed uniformly on the paths.

Given a trajectory $\mathcal{P}$ (a curve in the plane), we adopt a parametrization (e.g., uniform, but other parameterizations may also be used, as will be shown later) and take k samples $\{p_1, p_2, \cdots, p_k\}$, on $\mathcal{P}$. We define the distance between two paths $\mathcal{P}, \mathcal{P}'$ as $\pi(\mathcal{P}, \mathcal{P}') = \sum_{i=1}^{k} ||p_i - p'_i||^2$, the sum of squared distance between corresponding sample points. From a different viewpoint if a path $\mathcal{P}$ is considered a point in $2k$ dimensional Euclidean space $p = (p_1, p_2, \cdots, p_k)$ (each point p_i is a point on 2D), the distance between two paths is the squared ℓ_2 norm of their corresponding representative $2k$-dim points. In the following we will see this observation become very useful in the design of a succinct data structure that summarizes the relative distances of a set of paths by a set of points on a 1-dimensional line.

D. Locality Sensitive Hashing

The memory foot-print of full path history on sensors would be too large. Consequently, we adopt locality sensitive hashing

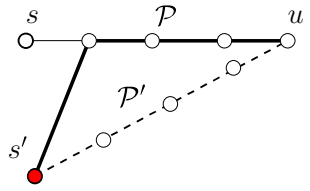

Fig. 2. The real path $\mathcal{P}$ taken by a packet from s is different from the path $\mathcal{P}'$ it should have taken if it were generated from the claimed location s'.

[7] a mechanism perfectly suited to compress such data and represent each path by a single value. The distance between two paths then becomes the distance between their compressed values. In general, locality sensitive hashing takes points in high dimensional space and maps them to 1D such that the Euclidean distances between them are roughly preserved. Recall that each path can be considered as a point in $2k$-dimensional space, which is then hashed to a point in 1D such that the distance between any two paths is correlated to the distance between their corresponding 1D points.

Locality-sensitive hashing makes use of the properties of stable distributions. A stable distribution [4] is a distribution where the random variable $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{1/p} X$, where $X_1 \ldots X_n$ are i.i.d. variables from that distribution. It is known that Gaussian distribution is stable for ℓ_2 norm. This means that if we represent a path $\mathcal{P}$ in our network by a vector in $2k$-dimensional space $v = (p_1, p_2, \cdots, p_k)$ and generate a random vector a (with each element chosen uniformly randomly from a Gaussian distribution $\sigma(0, 1)$), of the same dimension, then taking the dot product of the two vectors, $a \cdot v$, results in a scalar distributed as $||v||_2 X$, where $|| \cdot ||_2$ is the ℓ_2 norm, and X is a random variable with Gaussian distribution $\sigma(0, 1)$. It follows that for two vectors (v_1, v_2) the distance between their hash values $|a \cdot v_1 - a \cdot v_2|$ is distributed as $||v_1 - v_2||_2 X$ where X is a random variable of a Gaussian distribution. Therefore, if we have a vector v_i, which represents a path in our network, we can generate a scalar value from it (by taking the dot product with a) that still maintains the property that its distance from another scalar generated by another vector v_2 is correlated to the original "distance" between v_1 and v_2 as we previously defined. A hash function that uses random variables of a stable distribution to map high-dimensional vectors to 1D points satisfies the above definition of locality sensitivity.

Upon receiving a packet with observed trajectory vector v, each sensor will use locality sensitive hashing to store only the hashed value $h(v) = a \cdot v$ together with the location where this packet was generated. For a new packet that claims to be from the same region, the hashed value of the new packet is compared with the hashed values in the past history. A belief rating directly proportional to the difference of these two values is then generated.

In practice, to reduce overhead, a packet will only carry the position of the last node visited and the hash value computed by that node. A new node will update the packet with its own location and the new hash which is computed as the sum of the old hash plus this node's $h(v)$ value. This yields similar results

to the method described above, while significantly reducing communication overhead.

E. Belief Generation

By the property of locality sensitive hashing, packets taking paths similar or identical to each other tend to cluster together on the real line, while packets coming from unknown regions or following a highly irregular path map to different points. To express a correctness belief about the claimed location of incoming packets, their associated hash values are thus compared with the expected value ranges of nodes originating from the same region.

V. Security

We now describe the additional measures that must be taken to ensure that an adversary, or a group of adversaries working together, does not tamper with the belief ratings or hash values of packets passing through it. We want to ensure that (i) individual belief values are not tampered with in transit, (ii) packets containing incriminating ratings cannot be distinguished from other traffic, (iii) new fake belief values are not added to bad-mouth a packet or improve its rating, and (iv) existing belief ratings are not removed from packets.

A. Semantic Security

To ensure the above, we first require that each sensor be associated with a unique, public identifier (e.g., MAC) and with a secret, unique symmetric encryption key, known only to a very small set of authorized, un-compromised parties such as the data sink or a few intermediary relays, called packet evaluators, PEs. This is a reasonable, practical assumption to be found in existing research [2].

This key can then be used for communication between the sensor and the PEs. Such communication however, we require to be deployed using any semantically secure encryption cipher [5] (e.g., any cipher running in CTR mode). Semantic security is necessary to prevent an adversary of correlating encrypted fields in the current packet with fields of previously seen ones (e.g., if they represent the same value). Our solution does not depend on the deployed encryption mechanism. Symmetric key cryptography has been chosen over public key crypto, due to the computation-limited platform assumed. While details are out of scope here, we note that more powerful mechanisms can be devised using asymmetric key primitives. Such mechanisms would allow optimized, in-network location claim evaluation and packet filtering, effectively reducing overhead induced by compromised traffic.

B. Secure Belief Propagation

Upon generating a belief b (composed of a rating and confidence, see section IV-E) and hash for the current packet, a sensor i will encrypt it using its shared symmetric key with the sink k_i and append the result $E_{k_i}(b)$ to the packet. Requirements (i) and (ii) above are naturally handled. To ensure (iii) and (iv) we propose to use a cryptographic digest[1] chain constructs. Specifically, upon propagating a new belief b with the current packet, a sensor i will perform two operations. First it will encrypt the belief as above $E_{k_i}(b)$. Second, it will update the packet's digest chain value c as follows:

$$c_{new} \leftarrow H(c_{old}|H(b)),$$

where H is a cryptographic digest, and '|' denotes concatenation. At the PEs end, each packet's digest chain can be reconstructed and verified upon decrypting all beliefs. To selectively remove a rating from the packet, a malicious adversary is faced with having to correctly reconstruct a new digest chain for the remaining beliefs. This, however, will require the decryption of those beliefs (using secrets not in the possession of the adversary). Thus (iii) and (iv) are handled. An adversary can still attach a bad belief rating, i.e., bad-mouth a packet. But this is essentially denial of service attack in which the relay adversary can simply drop the packet from the data stream. As message digests over small amounts of data are extremely fast, the induced overhead is minimal.

To summarize, a node receiving a packet will append an encrypted belief rating, update the digest chain as mentioned above and deliver the packet to the next hop. The way that nodes attach belief ratings can be adaptive to the network scenario and desired detection ability. We now introduce two belief generation and propagation methods.

VI. Overhead Analysis

Communication. The number of belief ratings that is propagated with packets is network specific, but in usual scenarios we estimate 5-6 such values (5-6 bytes total). Other protocol-related information each packet carries is the location of the previous node on the path (2 bytes), the cryptographic digest used to protect the belief ratings (6 bytes) and the hash value computed by the previous node on the path (2 bytes). This would yield a total of about 15 bytes. Given that TinyOS packets in modern applications range anywhere from 36 to 100 bytes [1], this is certainly acceptable, especially in hostile deployment scenarios where such assurances are required.

In practice we further reduce the overhead by attaching beliefs in a probabilistic way. Only an adaptively small fraction of packets are randomly selected to carry beliefs. Alternately, only a small fraction of relay nodes attach belief ratings. Moreover, in a streaming application scenario, spanning multiple packets, only the first (header) packet in a data stream will need to carry this additional information—thus amortizing the overhead over the entire sequence.

Storage. The storage overhead is determined by the granularity of history traffic patterns. As we do not explicitly store all the hashed values of packets that visit a node, but rather

[1]We use "cryptographic digest" to denote a cryptographic hash function, so as to avoid any confusion with the locality sensitive hash constructs. We note that the collision resistance of such a hash function is not paramount here, given that the adversary would have a hard time finding meaningful collisions within a few minutes (and for a large enough number of packets to become meaningful), before the packet is due at the sink. We assume the hash function outputs 6 bytes.

only the mean and standard deviation of the values inside each quad, this results in a limited overhead. The amount of data a particular node stores is dependent on a its location within the network. A node near the center might be involved with messages passing through from multiple directions, while a node on the periphery will see data originating from only a few directions. This impacts the degree to which a node is able to group hashes of similar areas together.

For example, a maximum depth of the quadtree of 4 proved sufficient in our simulations to provide a partition of $1/256th$ of the network, a very detailed division of the sensor field. For a node at the center of the network potentially a full quadtree of depth 4 will be needed, with $4^4 = 256$ leaf nodes. A node on the periphery with only detailed information on half of the network, and no detail on the other half may have only $2 * 4^3 = 128$ leaf nodes. Similarly when nodes only route data to and from a few (1 or 2) sinks, the quadtrees of all nodes won't have more than 128 partitions. Each leaf node requires 3 bytes to store the mean (1 byte), standard deviation (1 byte), and the number of hashed values (1 byte). For a quadtree with 128 leaves, this requires 386 bytes of memory. Moreover, we can further reduce this overhead by adaptively considering lower depths in (un-interesting sub-branches of) the quadtrees, depending on available memory.

VII. Experimental Results

We validated our model by simulation with networks of various sizes, ranging from 50 to 1000 nodes. We considered geographic routing (GPSR [10]) to route messages between nodes and data sinks. We note however that our solution is not hard-coded to GPSR and works with other routing protocols. To model link and node failures, links and nodes are set to go offline at various times – links were active only 85% of the time and nodes 95% on average. Unless where otherwise noted, we used a single sink located at the center of the network, at $(x_{w/2}, y_{w/2})$, where w is the width of the network.

The network was first trained to understand the traffic pattern, by assuming a short interval of uncompromised traffic. In this interval, nodes normally pass messages to the sink, while observing routed traffic and building the hash history. We were then able to test the effect of moving one node to another location, and compare how the distance moved relates to our ability to detect it.

A. Model Validation

Figure 3 illustrates that the further away the claimed location of a node is, the greater the change to its hash value. The figure plots the change in hash value with respect to the change in path "distance". The x-axis shows the difference between the honest path and dishonest path by computing their "distance" as previously defined — by summing the distances between all the sample points of the 2 paths. For this simulation a network of 50×50 was used with a communication radius of 10 and a sampling rate of 100. A node was randomly selected at distance at least d from the sink, d being the width of the network. False locations were acquired by randomly choosing a direction and calculating the coordinates of the new claimed location, based on the magnitude of the claim. The strong linear correlation shows that our distance measure between paths truthfully captures the severity of the wrong location claim.

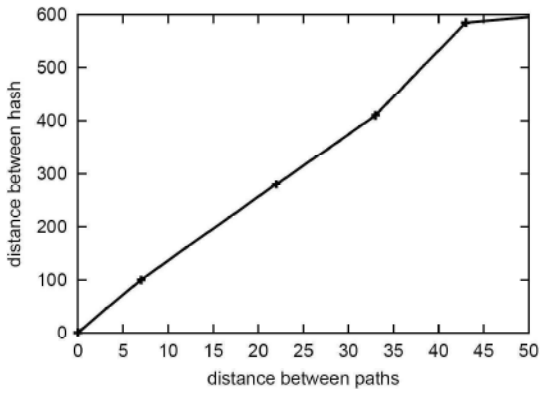

Fig. 3. As the distance between claimed and true location increases so does the difference between resulting hash values.

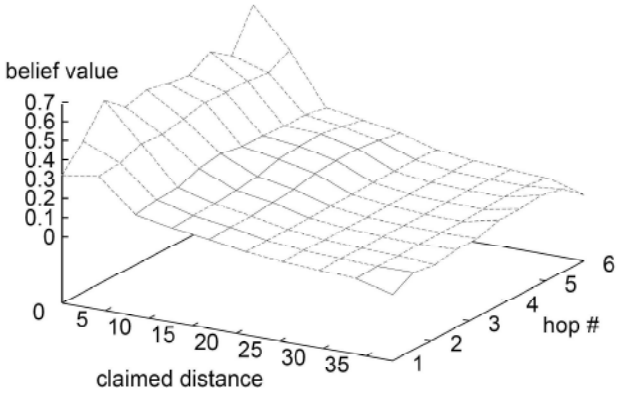

Fig. 4. The beliefs generated by each node along the path, as the adversary claimed distance increases.

Figure 4 shows the actual belief ratings formed by each node along a 6 hop path from source to sink. Values are shown for when the adversary claims to be at positions from 0 to 35 units away away from its real location (or 0 to 4 hops as the nodes have a radius of 10) in a network of 50×50. As can be seen, the belief quickly drops once the adversary makes a claim more than 1 hop away.

B. Parameter Fine-tuning

We mentioned how the hash function requires a random vector of size $2k$ where k is the degree of parametrization of the path ("sampling parameter"). Figure 5 shows that even a low sampling parameter will result in quite accurate belief ratings. There is no clear distinction between a value k ranging from 20 to 2000, only the sampling parameter of 10 appears insufficient. This is important, because, by using a lower parametrization we reduce the overhead of the hash function computation. This graph only gives a snapshot of what one

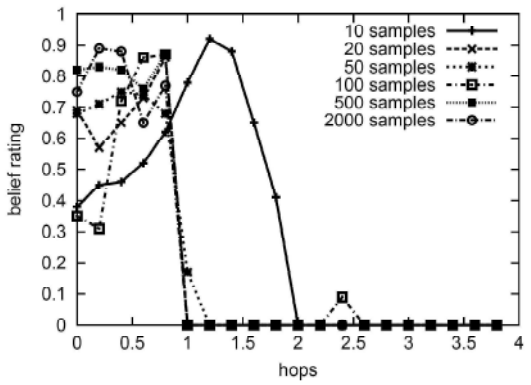

Fig. 5. Belief values (as a function of hop-distance of claimed vs. true location) using various samplings of the paths. This shows that we can reduce the overheads of hash function computation by using a lower parametrization.

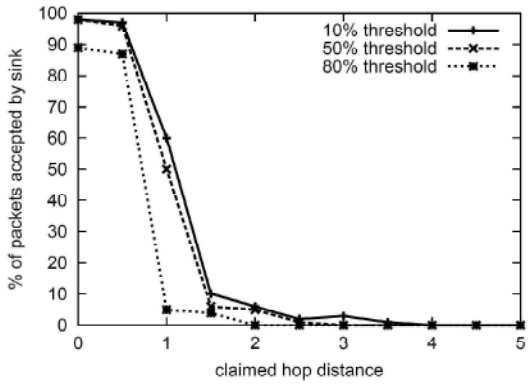

Fig. 6. The percentage of the number of packets accepted by the sink node with respect to distance claim of node in terms of the number of hops away it is from true location. The beliefs received at the sink must be above the given threshold value to be accepted.

node "believes". A node on the path from source to sink was arbitrarily selected (in this case the node at hop 4) and it's beliefs were plotted, this being the reason for the routing irregularities seen in the figure.

C. Detection of malicious claims

Figure 6 illustrates the percentage of packets accepted by the data sink after examining their belief values and comparing them with a "threshold value" for acceptance. Specifically, the sink will require the average of the lowest 3 beliefs on the path to be above the threshold to be accepted. The graph also shows the percentage of honest nodes accepted (the distance between claimed location and the true location is 0). Having a high threshold of 80% is too strict, and some honest packets are therefore incorrectly dropped. For this simulation we incorporated variability into the routing pattern by having only 85% of the links be active at any given time (thus the routes taken by the packets from the same source vary).

The results show that having only a small percentage of nodes attach beliefs is sufficient for a surprisingly strong detection ability. For example, by just having 5 beliefs attached, the detection accuracy is 95%.

VIII. Conclusion and Future Work

In this paper we have introduced a collaborative location certification scheme that determines whether nodes are falsely claiming incorrect locations in their event reports to the sink. Experiments have shown that with low overhead this scheme can detect incorrect location claims as close as one hop away from the node's true location.

References

[1] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. *Mobile Data Management (MDM)*, 2007.

[2] S. A. Çamtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical Report TR-05-07, Rensselaer Polytechnic Institute, March 2005.

[3] S. Cheung, B. Dutertre, and U. Lindqvist. Detecting denial-of-service attacks against wireless sensor networks. Technical Report Technical Report SRI-CSL-05-01, Computer Science Laboratory, SRI International, May 2005.

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on *p*-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press.

[5] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[6] Y. C. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM*, 2003.

[7] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM Press.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. pages 56–67, 2000.

[9] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, September 2003.

[10] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM Press.

[11] L. Lazos and R. Poovendran. SeRLoc: secure range-independent localization for wireless sensor networks. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 21–30, New York, NY, USA, 2004. ACM Press.

[12] Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust statistical methods for securing wireless localization in sensor networks. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks*, pages 91–98, 2005.

[13] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, 2002.

[14] K. Sanzgiri, B. Dahill, B. Levine, and E. Belding-Royer. A secure routing protocol for ad hoc networks. In *International Conference on Network Protocols (ICNP)*, November 2002.

[15] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *In Proc. of WISE 2003*, 2003.

[16] B. Waters and E. Felten. Secure, private proofs of location. Technical Report TR-667-03, Princeton University, January 2003.

[17] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.

[18] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 28, Washington, DC, USA, 2003. IEEE Computer Society.