2018 IEEE 37th International Symposium on Reliable Distributed Systems

Adversarially-Resistant On-Demand Topic Channels for Wireless Sensor Networks

Hans Walter Behrens
Arizona State University
Tempe, Arizona 85281–3673
Email: hwb@asu.edu

K. Selçuk Candan
Arizona State University
Tempe, Arizona 85281–3673
Email: candan@asu.edu

***Abstract*—Wireless sensor networks and other power-efficient devices fill increasingly important roles in modern society. At the same time, they also face increasing internal and external threats, such as node capture or protocol disruption by adversarial agents. Providing reliable and secure service in the face of these challenges remains an ongoing problem, and one that is only exacerbated by the computational and power constraints imposed on these devices. In this paper, we first introduce the concept of on-demand topic channels in the context of ephemeral wireless sensor networks. Then, building on this concept, we introduce three novel messaging protocols to provide secure, authenticated communication between a sensor network and an authorized user while also providing resilience from accidental or adversarial disruption. These protocols leverage homomorphic hashing in innovative ways to trade secrecy against network and computational costs in on-demand topic channel authentication. Finally, we compare and contrast the costs of these protocols, and show that hash-based protocols provide significant implementation-independent improvements to network resilience.**

I. Introduction

The Internet of Things (IoT) refers to the networks of low-cost, low-power devices used to sense and manipulate the physical world. As energy efficiency increases, and cost decreases, the proliferation and applicability of these devices continues to increase.

A. Ad-Hoc On-Demand Sensors

Although types of such devices are wide-ranging, we will focus out attention on one particular sub-type, wireless sensor networks (WSNs) supporting *on-demand* queries. These are spatially-distributed deployments of low-powered devices [1], which collectively sense their environment and record or report that information to a user, system, or database over a wireless medium upon request. In contrast to many IoT applications, these do not generally manipulate their surroundings directly. Since these devices' primary responsibility lies in the collection and delivery of this data, and not in transformation or analysis, the computational and power requirements of these devices are much lower than traditional computers. Although the network topology of such sensors may vary, *ad hoc* or *mesh-based* deployments are quite common, as these require neither precision placement of sensors nor powerful coordinator nodes to support evolving network topologies [2].

Funding for this research was provided by the National Science Foundation under the proposal "DataStorm: A Data Enabled System for End-to-End Disaster Planning and Response" (NSF Award No. 1610282)

Due to these devices' relatively low costs, they have broad appeal in solving a wide variety of problems. Industrial process monitoring, for example, can be simplified with robust sensors placed in areas where human monitoring would be difficult or dangerous. Simple logistical trackers permit long supply chains with very little loss, at a scale impossible for manual methods. Low-cost sensors permit users to track real-time changes in their environments in ways not previously possible.

B. Case Study: Disaster Response

The application of IoT systems for disaster response provides a good case study: the use of such systems has been extensively studied in the literature [3], [4], [5], but significant questions remain unanswered.

1) Opportunities: The deployment of ad hoc devices in response to natural disasters provides emergency personnel with a much better situational understanding in areas where underlying infrastructure may be damaged or nonexistent. WSN deployments in this context are intrinsically ephemeral, requiring deployment in challenging environments while also providing reliable service for the duration of the response. In many cases, disasters cannot be predicted early enough to allow proactive deployment of sensor networks; instead, emergency responders requiring WSN coverage must rely on reactive deployment. Since existing power and communication infrastructures may be damaged, destroyed, or nonexistent, any sensor network should be entirely self-contained to maximize deployment flexibility. Once deployed, these finite power resources must be managed efficiently to ensure that services can endure throughout the critical response period. Currently, disaster response relies heavily on sensing satellites to fill this requirement [6], but high costs, delayed deployment, and coarse sensing resolution suggest an alternative approach could offer substantial benefits. Localized deployments of comparatively high-resolution WSNs mitigate satellites' associated disadvantages, though they are also more susceptible to power and communication disruptions on the ground than space-based platforms. WSN deployments may be achieved directly by emergency personnel, over a large area via aerial dispersion, or by drone deliveries.

2575-8462/18/$31.00 ©2018 IEEE
DOI 10.1109/SRDS.2018.00019

2) Challenges: Damaged infrastructure may be effectively bypassed through the use of low-cost, battery-powered nodes communicating over an ad hoc wireless network, but the advantages offered by self-contained WSNs come with additional costs. Constraints on size, complexity, computational resources, and battery life must now be considered. As physical size increases, deployment logistics become more complicated, potentially restricting available delivery methods. As computation and wireless communication needs increase, power drain on the devices' batteries also grows; once a device has been depleted, it may be logistically impossible to recharge or refurbish that unit. Additionally, sensors may intermittently or permanently fail for a number of reasons. Beyond battery depletion, nodes may experience unplanned hardware failure, or an interruption in communications with neighboring nodes. Criminals seeking to exploit a disaster may serve in an adversarial role, attempting to subvert or compromise sensor coverage. The deployed network should therefore be resilient, and remain available for readings despite such failures or attacks. To mitigate the risk of failures, many deployment strategies call for the inclusion of redundant nodes [7], so that some proportion of the participating nodes can fail before network degradation begins.

Furthermore, disaster response activities may necessitate layered, heterogeneous deployments of different sensors. Search and rescue operations, infrastructure monitoring, damage assessments, or warning systems may each require different sensor types, spatiotemporal data resolutions, or security requirements. Despite this wide range of needs, logistical costs associated with delivery may constrain responders to very few, or even one single deployment to cover all these demands.

Emergency responders may need to query different aspects of this heterogeneous deployment relevant to their team's goals, without impacting the activities of other teams by unnecessarily depleting node resources, or sifting through non-relevant data in time-sensitive situations. Due to the diverse nature of these IoT deployments, various streams of information may be available to responders in a disaster scenario. However, not all data will be relevant to all responders; by compartmentalizing information and releasing only to users who are both relevant and authorized, operational security and network efficiency can both be improved. We use the term *topic channels* to refer to these semantically-grouped heterogeneous sets of sensors. Although some preliminary research in this area has been conducted [8], topic channels have thus far been paired with publish-subscribe frameworks, in an event-driven approach that does not take changing user demand for information into account. By adapting such topic channels to an on-demand paradigm, the number of sensor readings and transmissions by the network may be drastically reduced, security vulnerabilities prevented, and total network service life extended.

C. Contributions of this Paper

Given this context, we aim to answer the following questions: (1) Can we provide a protocol supporting on-demand querying of topic channels within a heterogeneous, ad hoc sensor deployment? (2) Can the security of our protocol be guaranteed against a wide variety of attacks and adversarial scenarios, while also maintaining network availability and robustness throughout? (3) Can our protocol support post-deployment modifications to topic channel assignment, without compromising the security of the network? (4) Can the per-node efficiency of our protocol be increased through the use of less computationally-intensive approaches? (5) Can we minimize inter-node message transmission to reduce our network's total energy usage? To address these challenges, we propose and compare three novel network messaging protocols that support adversarially-resistant on-demand topic channels:

- The first protocol is a novel hash-based protocol, leveraging multicast communication to simplify delivery. This approach combines the security and efficiency benefits of standard encryption protocols, but sacrifices the secrecy of the messages. However, limitations on response authentication in this protocol may leave the network open to exhaustion attacks during the response phase.
- The second protocol introduces the concept of channel validation, which leverages on-node computations by channel members to mitigate the impact of falsified responses sent by adversarial nodes.
- The third protocol builds upon the previous ideas by introducing chain validation, which uses structural properties of the network to constrain message propagation, further reducing power consumption during the response phase.

The paper is organized as follows: Related works are introduced in Section II. The system model and assumptions are provided in Section III. We describe the protocols in Section IV, followed by their assessments in Section V. We conclude in Section VI.

II. Related Works

A. Topic Channels

The publish-subscribe model [9] describes a system in which streams of information are produced by participating data sources. Data consumers can choose to consume these streams of information based on the topics they are interested in; therefore, these streams are sometimes referred to as *topic channels*. A key point of these systems is that data is generally produced independently of the number of subscribers; that is, it is fundamentally event-driven, or push-based. In applications where real-time perception is desired, this is an ideal model. For example, [8] explores the applications of this technique in low-powered deployments within the disaster response context.

Conversely, in some applications, data is not collected or generated except when it is requested, or pulled, by the consumers. To differentiate this inverted approach, which could also be termed "subscribe-publish", from more commonly-understood meanings of publish-subscribe and prevent confusion, we refer to this as "on-demand" or "query-response".

This approach is especially beneficial when the number of requests is greatly outnumbered by changes in the sensed data, as it dramatically reduces the amount of communications over the network. We leverage this second model to increase deployment longevity by reducing message transmissions, and to strengthen security by preventing compromised nodes from communicating across the network except in response to valid queries, thereby preventing certain types of DoS attacks.

B. Homomorphic Hashing

In this paper, we leverage the class of collision-resistant homomorphic hash functions, which are one-way functions mapping inputs into fixed-size outputs with the additional property that their application is homomorphic; for example, given a function $H(x)$, we can say that H is homomorphic if $H(a)|H(b) = H(a|b)$. These functions have been described in the literature by Krohn et al. [10] through the application of the discrete logarithm problem, but other constructions, such as the elliptic curve multiset hash [11], may also be used. We leverage this class of functions to increase the security, flexibility, and efficiency of our proposed protocols.

C. Hashing Efficiency

Hashing is computationally more efficient on low-powered devices than either message authentication (MAC) or encryption. Although special instructions and optimizations exist to make encryption more efficient on certain chips, these are rarely present on very low-powered devices, such as the sensor nodes used in WSN deployments. These conclusions are supported by the findings of Pereira *et al.*, in their comprehensive benchmarking of WSN-suitable chips for these three classes of functions [12]. Note that this work did not explicitly evaluate homomorphic hash functions, whose performance characteristics may differ from more traditional collision-resistant one-way functions.

Given the relative efficiency of hashing in the context of low-powered devices, it is not surprising that the use of hashing to increase performance and longevity of ad hoc networks has previously been explored. [13] makes use of variable rounds of hashing to establish secure session keys for encrypted communication between neighboring nodes; this contrasts with the approach proposed in this paper, which makes use of deterministic hashing rounds, without encryption. Additionally, once a node in their model has been compromised, it gains the ability to compromise message integrity or exhaust the available resources of the network as a whole, since there is no centralized source of authority.

[14] examined the applications and efficiency advantages of homomorphic hashing for IoT networks in the context of message aggregation. In this application, messages are aggregated within the network itself, using homomorphic cryptographic primitives to secure this process and reducing the number of messages which must propagate back to the base station. Their approach provides several benefits over previous methods, but does not support on-demand topic channels.

D. Adversarially-Resilient Messaging

Other examinations of adversarially-resistant message delivery and authentication have also appeared in the literature. [15] approaches the problem using cryptographic primitives with a strong adversary, but does not address the problem of node capture attacks.

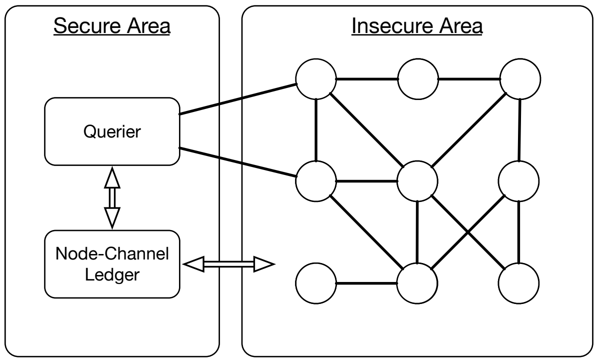

Fig. 1: Example network model

Message routing itself provides an attack surface for adversaries to exploit. [16] proposed Ariadne, a distributed protocol for securing message routing for ad hoc networks. However, their approach is restricted to the routing messages themselves, which are used to 'learn' the topology of the network. Additionally, it requires either time synchronization, which is rarely possible in low-power deployments, or shared secret keys, which are vulnerable to node capture attacks.

III. Problem Formulation

Before describing our protocols in detail, we first establish the constraints and assumptions that the protocols will obey.

A. Network Model

As seen in Figure 1, an agent in our network can take one of three possible roles: a *node-channel ledger*, a *querier*, or a *sensor node*. We assume that the node-channel ledger and the querier are not resource constrained, as might be represented by a server or local workstation. In contrast, a sensor node has significant computational and power constraints.

1) Topology: We further assume an implementation-agnostic, graph-like network model, in which each agent may communicate directly with its neighbors, or indirectly with one or more other nodes through multicasting. However, we do not specify the underlying communication or hardware requirements; we assume the network supports flood-based stateless multicasting, in which the overall topology is not learned by the constituent nodes. We consider this case to be the worst-case scenario for multicast implementations; more-efficient approaches [17], [18] may further increase the efficiency of our multicast-based protocols by reducing redundant communication incurred by message broadcasts.

2) Communication Medium: Although the device network could be either wired or wireless, the former is very rarely seen in ad hoc sensor deployments, and we therefore assume agents communicate wirelessly. A common characteristic of wireless communication is that transmitting messages is significantly more expensive than receiving messages. We therefore primarily attempt to minimize the number of message transmissions by non-compromised sensor nodes.

3) Ephemerality: All sensor nodes in the network are considered *ephemeral* and *fragile*. Ephemerality describes nodes which may temporarily drop out of the communication network, and reappear later at their previous location; for example, a temporary disruption of service corresponding to network interference, or a power-saving sleep cycle. Fragility is used to describe the fact that nodes may drop out of service unexpectedly and permanently; this could correspond with a depleted battery or an unrecoverable hardware error in the device. Although these may lead to network partitioning, the proposed protocols can efficiently support partitioned networks with very little efficiency loss[1].

4) Secure vs. Insecure Areas: We specify two different regions of the network, the *secure area* and the *insecure area*. The secure area corresponds to a well-protected location within the deployment area, such as a field office; the insecure area corresponds to the field deployment itself and represents the area that cannot be reasonably protected against intrusion by an adversary. All sensor nodes deployed in the network are assumed to be located in the insecure area, while both the querier and the ledger are assumed to be in the secure area. We further assume that all communication between participants, regardless of location in a secure or insecure area, is susceptible to our adversarial threat model. However, participants that are located within a secure area cannot be directly compromised by the adversary (either physically or remotely). Conversely, the adversary is able to compromise existing nodes anywhere within the insecure area. Additional capabilities are detailed in Section III-C.

5) Entrance Nodes: We also specify the concept of *entrance* or bridge nodes. These represent sensor nodes (usually at the periphery of the deployment) that serve as gateways for messages transitioning between the secure and insecure areas. Larger number of entrances reduces the likelihood that the network will be partitioned such that one of the subgraphs is unreachable to either the ledger or the querier. Furthermore, more entrances improves "wear leveling" on these nodes, as messages being transmitted back to the querier may transition through any of the querier's entrance points, rather than forming a transmission bottleneck on only a few nodes. Entrance redundancy also ensures that if the adversary compromises one of these nodes, it is less likely to be able to sever communication with the network.

B. On-Demand Sensor Networks

1) Topic Channels: Given a set of heterogeneous sensor nodes $\mathcal{N}$, with each member N_i containing a set of distinct sensors $\mathcal{S}_i$, we define the concept of a *on-demand topic channel* (ODTC) T as a subset of the set of possible sensor types $\mathcal{S}$, where $\mathcal{S} = \bigcup_{i=1}^{|\mathcal{N}|} \mathcal{S}_i$. We also introduce the concept of a *response rate* to a given ODTC T. This describes the percentage of total nodes in $\mathcal{N}$ which have sensors that are members of T. Note that this explicitly does not correspond to the number of nodes which *actually* respond to a query against a given ODTC, for example due to our concepts of ephemerality and fragility, but to the nodes which *should* respond if all such nodes were functioning.

[1]Discussion of the proposed protocols' behavior under partition is beyond the scope of this paper.

2) Node Deployment: Each node is assigned a unique identifier, such as a UUID [19], prior to being deployed into the field. This simplifies the job of the ledger in tracking which secure communication channel is associated with each node, and permits the ledger to address communications directly to specific nodes. Each item in a node's sensor suite is assigned a simple numeric ID, for later use when assigning topic channels. The capabilities of a node, in terms of sensor count and capabilities, must be known to the ledger in order for sensor-topic mappings to be created. Alternatively, if assignments by overall node class are preferred, then distinct node hardware configurations can be considered as single sensor types, and assigned to various topic channels as appropriate.

New sensor nodes may be added into the deployment at any time. To support this, there must be an existing encrypted channel to communicate securely and secretly with the deployed node, for preliminary handshaking. Generally, this is accomplished by either loading a key during manufacturing, installing a key via physical access prior to deployment, or by using a session key-based communication protocol. Securing this channel is outside the scope of this work, and existing approaches may be adopted to satisfy this prerequisite [20].

Since use of an encrypted link may be more expensive to use relative to our hash-based approaches, we aim to minimize messages along this channel, primarily reserving this method for channel assignment or other sensitive administrative tasks.

3) Querier Authentication: When a querier wants to conduct a query against a ODTC, they must first contact the ledger to request permission to query that channel. This exchange should be conducted securely, as the data exchanged may contain sensitive information which would compromise the network if leaked; for simplicity, this paper assumes an existing secure two-party communication protocol is used. A successfully-authorized querier provides to the ledger a set of one or more topic channels to query, as well as its own credentials (if any) for authentication. Upon receipt and validation of a query request, the ledger returns to the querier the necessary information to issue one query; this process is described in further detail in Section IV-B4.

4) Messages & Requests: To facilitate message delivery, we will consider two possible network knowledge models. The first assumes that the network topology and changes to it are known in real time to the querier and ledger. We refer to this method as the "optimistic" or "known" model. In the second case, we assume a true ad hoc deployment, where the network topology (except for an agent's immediate neighbors) is unknown to the participants, nor can it be efficiently learned by the nodes. We refer to this as the "pessimistic" or "ad hoc" model. We also describe four potential message delivery methods, used in different phases of our proposed protocols.

a) One Message Direct Routing (OD): This approach describes the process of a single message traversing the network between two agents, where the message recipient is predefined. In the case where the network topology is known, and the transmitter is not computationally bound, we also allow the shortest path through the network to be computed. Otherwise, when the topology is not known, direct routing is only possible under paradigms in which the network topology and routing rules are learned; we do not assume such an approach.

b) N Message Direct Routing (ND): This approach describes the process of delivering distinct messages from one source to multiple recipients. In a general case, this approach can be thought of as an extension to OD, iteratively repeated proportionate to the number of unique messages.

c) One Message Multicast Routing (OM) : This approach describes a single message being multicast to all possible recipients. Each node which receives this message will store a copy locally for potential on-node processing, and will then forward the received message to all neighbors except the neighbor from which it received that message. Other more efficient ad hoc multicast implementations, such as those described by [21], may also be used.

d) N Message Multicast Routing (NM): This approach describes the process of multiple, distinct messages, each being delivered to multiple recipients. In a general case, this approach can be thought of as an extension to OM, iteratively repeated proportionate to the total number of messages to be delivered.

C. Threat Model

We use the NIST 800-154 threat model [22] to describe network security, with the following potential threats:

- *Eavesdropping*: The adversary may listen to and record any message sent on the network, i.e. through the deployment of a very sensitive omnidirectional antenna.
- *Message injection*: The adversary may send messages from a compromised node at any time, to any valid recipient. The number of transmitted messages is constrained by the sending node's power budget. This serves as the primary available interaction to disrupt the network.
- *Black hole attacks*: The adversary may block (i.e. fail to forward) messages traversing routes which pass through an adversarially-controlled node.
- *Node capture attacks*: The adversary may gain control over a given node, either through physical means such as flash dumping, or remotely via exploiting a software vulnerability. In either case, the adversary gains all knowledge known by the compromised node.
- *Replay attacks*: The adversary may re-transmit one or more previously-recorded messages between parties on the network, including valid requests from a querier or responses from sensor nodes.
- *Denial of service attacks*: The adversary may attempt to disable the functionality of the network through the broadcast of malicious messages, either to flood the network with irrelevant information, or to overwhelm the power budgets of the participating sensor nodes.
- *Masquerade attacks*: The adversary may impersonate one or more sensor nodes in the deployment, returning misleading or unintelligible results to the querier.

All data, including both protocol and sensed information, is stored on-device in either RAM or disk storage and is therefore susceptible to adversarial capture on devices located in the insecure area. This data moves through the system wirelessly, using protocols as described in Section IV. Securing the wireless transmission itself, or device security to reduce risk of compromise, are outside the scope of this work but could serve to augment the security of the proposed protocols.

We assume the adversary may transmit a similar message at the same time as a valid one, but cannot remove or otherwise modify the valid message; the impact of wireless jamming on the network environment is beyond the scope of this work. The objective of the adversary is to disrupt the functionality of the deployment, for example by spoofing node responses to return false information to the querier, or to prevent the querier from collecting relevant results from the network. The defender aims to maintain sensing function despite intentional attack or unintentional degradation.

Finally, we consider the querier as a 'trusted-but-curious' participant. They will obey the protocol and follow all constraints, but may attempt to access information outside of their authorized scope if able to do so. This could include, for example, a curious low-level employee with restricted access to only a few sensors deciding to instead query all topic channels in the network.

IV. Authentication Protocols

In this section, we present three on-demand topic channel authentication protocols for ad hoc IoT networks using several conventions; here, S_i denotes the i^{th} item of a set $\mathcal{S}$, $N.U$ denotes a characteristic U of an object N, $A|B$ represents an operation on A and B which is closed under homomorphism, and `function()` denotes a subroutine.

A. ***Baseline: Encryption-Based Protocols***

Many secure communication protocols, including those for ad hoc systems, make use of encryption to provide message authentication and secrecy. In general, these approaches may take one of two possible approaches: (A) a single encryption key is shared between all participating nodes, and is used to encrypt communication between all recipients, or (B) unique encryption keys are established for each communicating pair, and the appropriate key-pair is selected based on the sender and recipient of a message. These baseline protocols are undesirable for two reasons. First, the former approach is completely subverted through the node capture of a single node; once the shared key is compromised, the entire network can be disrupted. Second, the latter approach limits the opportunities for intermediate nodes to conduct adversarial mitigation, ultimately undermining network longevity. Our protocols aim to address both of these drawbacks.

Algorithm 1: Ledger creates new topic channel

```
Description: A new channel $T$ is generated by a local user on the
    node-channel ledger, with an associated label $L$, identifier
    $U$, and authenticator $A$, and stored locally in the ledger's
    set of channels $\mathcal{T}$.
Participants: A node-channel Ledger $C$.
Input: A local user interacts with the Ledger.
Result: A new topic channel $T$ is created.
1 Instantiate new channel $T$
2 $T.L \leftarrow$ channel label
3 $T.U \leftarrow$ unique numeric ID
4 $T.A \leftarrow$ randomly-generated string
5 $C.\mathcal{T} \leftarrow C.\mathcal{T} \cup T$
```

B. *Protocol #1: Query Validation*

Our first proposed protocol sacrifices message secrecy to increase on-node processing efficiency. We do this through the use of *query validation*, a method of certifying the authenticity of a query as it arrives at a node. This protocol uses the concept of request nonces, which are uniquely generated values associated with distinct transactions, and the concept of authentication strings, which are secret values associated with individual topic channels. Together, in conjunction with homomorphic hashing primitives, we are able to provide significant efficiency gains while only relaxing message secrecy relative to our encryption-based baseline protocols. The hash function, $\texttt{hash}()$, must (a) be collision-resistant; (b) provide results of a length sufficient to prevent an exhaustive evaluation; and (c) provide the homomorphic property. We do not explicitly require that $\texttt{hash}()$ supports keyed hashing; in cases where private information is needed to prevent attacks, we can include such information in the input itself.

1) Topic Channel Establishment: Topic channel establishment takes place on the node-channel ledger as described in Algorithm 1.

2) Topic Channel Assignment: To prevent impersonation attacks on the network, the authentication string for each node must be communicated secretly; if not, any node which intercepts the message will be able to masquerade as that node. This necessitates the use of encryption to preserve secrecy. Topic channel assignment is described in Algorithm 2. Additionally, pre-assignment of nodes to channels prior to deployment is also possible, through pre-loading of authentication hashes.

Algorithm 3 describes a node's assignment processing. Note that the ability to decrypt the smaller validation key is checked prior to the more-expensive decryption of the full payload. When a node N receives an assignment containing a mapping to a channel T, we say that N is *associated* with T. Prior to forwarding an assignment message, each node that belongs to at least one channel validates the assignment message using the signature of that channel, to ensure assignment messages are not replayed or falsified.

3) Topic Channel Updates: Since topic channel assignments are overwritten by each request, no explicit deletion or update requests are necessary. Changes for a given node's topic channel assignments can be included in a single request containing the appropriate new authorizations.

Algorithm 2: Ledger sends a new channel assignment

```
Description: A local user instructs the node-channel Ledger to assign a
    specific node to one (or more) topic channels. One
    constructed, these assignments are broadcast into the
    network addressed to the targeted node.
Participants: A node-channel Ledger $C$, and a sensor node $N$.
Input: A local user interacts with the Ledger.
Output: A new topic channel assignment message $M$.
1 Instantiate new assignment request $R$
2 $R$[validation] $\leftarrow (N.U, E)$, where $E$ is a monotonically increasing
    non-negative integer, maintained by the ledger, and incremented
    whenever any request is sent
3 $R$[assignmentNonce] $\leftarrow W$, a randomly-generated unique nonce
4 $R$[assignmentSequence] $\leftarrow E$
5 $R$[hashes] $\leftarrow \{T_i.L : \texttt{hash}(E|W|T_i.A)\}\ \forall i \in \mathcal{T}$
6 $R$[mappings] $\leftarrow \mathcal{C} = \{C_1, C_2, ..., C_k\}$ where $C_i$ is a 2-tuple of
    $(T_i.U, N_i.S_j)$ representing a channel ID and sensor type that should
    be associated
7 $R$[authorizations] $\leftarrow \mathcal{G} = \{(\texttt{hash}(N.U|T_i.A), \texttt{hash}(T_i.A|N.U))\}$,
    where each two-tuple in $\mathcal{G}$ contains an entry for each *distinct* channel
    $T_i$ in $\mathcal{C}$)
  // $M_d$ (below) only available under known network
     topologies.
8 $R$[metadata] $\leftarrow (M_d, M_t)$ where $M_d$ is an upper distance bound
    between any node and a bridge node, and $M_t$ is the expected
    maximum number of topic channels
9 Instantiate a new message $M$
10 $M$[type] $\leftarrow$ "assignment"
11 $M$[addressee] $\leftarrow N.U$
12 $M$[validation] $\leftarrow \texttt{encrypt}(N.U|E)$
13 $M$[payload] $\leftarrow \texttt{encrypt}(R)$
14 if the topology is known then
15     $\texttt{OD}(N, \texttt{serialize}(M))$
16 else
17     $\texttt{OM}(\texttt{serialize}(M))$
```

4) Query Processing: The first part of the construction occurs when the query request is made. Additional processing takes place on the ledger, which provides stronger control over the behavior of the querier, who must now contact the ledger before each query of the network. This process is described in Algorithm 4.

5) Response Processing: Once a query has been broadcast into the network by the querier, it is processed by a receiving node N as described in Algorithm 5.[2] When a node B receives a response, it processes the message according to Algorithm 6.

Since replies may arrive more than once due the multicast routing, the querier may wish to cache results according to sequence number and node ID in order to de-duplicate results.

6) Discussion: Through the use of homomorphic hashing, this protocol produces compound signatures, validating that the messages received by the participants did come from the claimed nodes, and that their content was not modified by the adversary. To accomplish this, several new variables are introduced, including nonce values for various phases of the protocol, as well as secret authentication strings known only to the ledger. By restricting the knowledge of these strings to the

[2]We assume that either each deployed node will know the network graph but be computationally-bound and therefore unable to compute the shortest path for direct messaging, or unable to discern the network graph at all. In either case, multicast responses (OM) are required. Since every node must return its own results to the querier, there are always multiple distinct messages to deliver; viewed in aggregate, this means that all protocols use NM to reply.

Algorithm 3: Sensor receives a topic channel assignment

Description: A sensor node receives an "assignment" message M. If addressed to N, the channel assignments are validated using encryption, and stored on-node; otherwise, the message is checked to ensure that it is not out of sequence and that it originated from the node-channel ledger before being forwarding to other nodes.
Participants: A sensor node N.
Input: An "assignment" message M.
Result: The receiving node stores channel authentication information.

```
if $M$[addressee] $== N.U$ then
    $M.V \leftarrow$ decrypt($M$[validation])
    if $M.V[0] == N.U$ and $M.V[1] \geq N.E$ then
        $P \leftarrow$ decrypt($M$[payload])
        if $P$[validation][0] $== N.U$ and $P$[validation][1] $\geq N.E$ then
            $N.E \leftarrow P$[validation][1] $N.\mathcal{T} \leftarrow P$[mappings]
            $N.\mathcal{G} \leftarrow P$[authorizations]
else
    if $M$[assignmentSequence] $\leq N.E$ then
        drop($M$)
    foreach $G_i \in N.\mathcal{G}$ do
        if hash($M$[hashes][$T_i.L$]$|N.U$) $\neq$ hash($M$[assignmentSequence]$|M$[assignmentNonce]$|G_i$[1]) then
            drop($M$)
    retransmit($M$)
```

Algorithm 4: Querier constructs a new request

Description: A Querier and a Ledger (indented) conduct a two-way interaction over a secure, authenticated channel to provide the Querier with information necessary to produce a single, valid query, allowing for mediation of access to channels.
Participants: A Querier Q and a node-channel Ledger C.
Input: A local user inputs a query on the Querier system.
Output: A new, valid query R.

```
secure-OD(Ledger, $\emptyset$)
    $\emptyset$ received                                   // Ledger $C$
    secure-OD(Querier, $\mathcal{T}$))
$\mathcal{T}$ received                                     // Querier
$\mathcal{T}' \leftarrow \{T_i\}$, a subset of $\mathcal{T}$.
$W \leftarrow$ a randomly-generated unique nonce
secure-OD(Ledger, $(\mathcal{T}', W)$)
    $\mathcal{T}', W$ received                             // Ledger $C$
    $\mathcal{H} \leftarrow \emptyset$
    $\mathcal{H}' \leftarrow \emptyset$
    $W' \leftarrow$ a new, randomly-generated unique nonce
    foreach $T_i \in \mathcal{T}'$ do
        $H_i \leftarrow$ hash($A_i|W|C.E$)
        $H_i' \leftarrow$ hash($A_i|W'$)
        $\mathcal{H} \leftarrow \mathcal{H} \cup H_i$
        $\mathcal{H}' \leftarrow \mathcal{H}' \cup H_i'$
    secure-OD(Querier, $(C.E, W', \mathcal{H}, \mathcal{H}')$)
$C.E, W', \mathcal{H}, \mathcal{H}'$ received              // Querier
Instantiate a new request $R$ $R$[type] $\leftarrow$ "query"
$R$[sequence] $\leftarrow C.E$
$R$[queryNonce] $\leftarrow W$
$R$[querySignature] $\leftarrow \mathcal{H}$
$R$[queryDetails] $\leftarrow \mathcal{T}'$
OM(serialize($R$))
```

Algorithm 5: Node replies to query, protocol #1

Description: A node receives a query request, and first determines if it should be forwarded. Any message associated to that node must pass query validation before forwarding. If N is in a queried channel, it then constructs a response, including a validation hash of the returned data.
Participants: A sensor node N.
Input: A "query" message M.
Output: A valid response R.

```
$\mathcal{R} \leftarrow \emptyset$
$\mathcal{T} \leftarrow M$[queryDetails]
$\mathcal{H} \leftarrow M$[querySignature]
retransmit $\leftarrow \varnothing$
foreach $T_i \in \mathcal{T}$ do
    if $T_i$ is associated with $N$ then
        retransmit $\leftarrow$ retransmit $\vee$ False
        if hash($N.G_{i,1}|M$[queryNonce]$|M$[sequence]) $==$ hash($N.U|H_i$) then
            $\mathcal{R} \leftarrow \mathcal{R} \cup T_i$
            retransmit $\leftarrow$ True
if retransmit $\neq$ False then
    retransmit($M$)
if $\mathcal{R} \neq \emptyset$ and $N.E < M.E$ then
    $N.E \leftarrow M.E$ $\mathcal{D} \leftarrow \emptyset$
    $\mathcal{V} \leftarrow \emptyset$
    foreach $R_i \in \mathcal{R}$ do
        $V_i \leftarrow$ hash($M$[queryNonce]$|D_i|N.G_{i,1}$)
        $D_i \leftarrow$ recorded datum
        $\mathcal{V} \leftarrow \mathcal{V} \cup V_i$
        $\mathcal{D} \leftarrow \mathcal{D} \cup D_i$
    Instantiate new reply $R$
    $R$[type] $\leftarrow$ "response"
    $R$[sequence] $\leftarrow N.E$
    $R$[queryNonce] $\leftarrow M$[queryNonce]
    $R$[node] $\leftarrow N.U$
    $R$[data] $\leftarrow \mathcal{D}$
    $R$[dataSignature] $\leftarrow \mathcal{V}$
    OM(serialize($R$))
```

Algorithm 6: Node receives reply, protocol #1

Description: A node receives a response to a query. If it is the Querier, it confirms the validity of the message. The Ledger drops all incoming messages, while intermediate sensor nodes forward such messages without additional validation.
Participants: One of: a Querier, a Ledger, or a sensor node.
Input: A "response" message M.
Result: The Querier validates and records the response, the Ledger drops the response, or the node forwards the response.

```
$\mathcal{D} \leftarrow M$[data]
$\mathcal{V} \leftarrow M$[dataSignature]
switch node type do
    case "Querier" do
        foreach $D_i \in \mathcal{D}$ do
            if hash($M$[queryNonce]$|D_i|M$[node]$|\mathcal{H}'$) $==$ hash($V_i|W'$) then
                include $D_i$ as a valid datum in the aggregation
    case "Ledger" do
        drop($M$)
    else
        retransmit($M$)
```

ledger, we constrain the querier – it is restricted to accessing only those topic channels for which it is authorized, and it may not query topic channels more than once without requesting new permission from the ledger. By using these hashing primitives to replace comparatively-expensive encryption functions, power consumption of the network can be reduced, increasing deployment longevity.

C. Protocol #2: Query + Channel Validation

Previously, queries were validated as they arrived, and responses were validated as they were delivered to the querier. However, this minimal level of verification leaves the network relatively unprotected against fabricated or falsified responses

Algorithm 7: Node replies to query, protocol #2

```
Description: As Alg. 5, except channel-related information from the
             query is also included in the response for later validation.
1  $\mathcal{R} \leftarrow \emptyset$
   [...]                                              // as in alg. 5
28     $R$[queryDetails] $\leftarrow \mathcal{T}$                        // Channel info
29     $R$[querySignature] $\leftarrow \mathcal{H}$                 // Channel signature
30     OM(serialize($R$))
```

Algorithm 8: Node receives reply, protocol #2

```
Description: As Alg. 6, with the addition of channel validation logic at
             the sensor node level. Responses which fail the channel
             validation are dropped from the network.
1  $\mathcal{D} \leftarrow M$[data]
   [...]                                              // as in alg. 6
11         drop $\leftarrow$ False                       // channel validation
12         foreach $T_i \in \mathcal{T} \wedge i \leq M_t$ do
13             if $\neg$drop $\wedge$ $T_i$ is associated with $B$ then
14                 if $hash(G_{i,1}|W|E) == hash(B.U|\mathcal{H}_i)$ then
15                     drop $\leftarrow$ drop $\vee$ (hash($W|D_i|N.U|G_{i,2}$) ==
                         hash($V_i|B.U$)
16                 else
17                     drop $\leftarrow$ True
18         if $\neg$drop then
19             retransmit($M$)
```

injected by adversarial nodes. Instead, these improper messages will be received and rebroadcast by the network, magnifying their power-consuming effects. To address this issue, we now introduce an additional layer of validation, which we call *channel validation*.

This approach permits constituent nodes to verify messages as they traverse the network, and to immediately drop those which can be recognized as false. Although additional computation is required to conduct such validation, substantial reductions in adversarial message transmission are possible, which may lead to an increase in overall network resilience.

1) Response Processing: Algorithm 7 includes channel-specific information in the query response, to permit nodes to determine if that reply is valid for that channel. This permits each node to serve as a gatekeeper for its own topic channels[3].

Algorithm 8 describes new validation processes introduced during the re-transmission phase. This prevents falsified messages from improperly responding to a query – if a datum's signature does not pass validation, then the reply is dropped.

2) Discussion: By further leveraging homomorphic hashing to validate messages as they propagate over the network, using an opportunistic approach, we are able to filter messages generated by adversarial nodes without requiring expensive setup. Instead, each node can make a local decision about the validity of the message as it arrives, and short-circuit the re-transmission if a problem is detected. However, valid messages are still sent to each node in the network, resulting in extraneous transmissions that unnecessarily stress the network's power capacity; this problem is addressed in the following protocol. Note that messages may still be correctly signed by the adversary using a captured node authorized for that channel. In this scenario, the sequence number could be leveraged to prevent such attacks from flooding the network.

[3]Any message must belong to a valid channel (i.e. T_i where $1 \leq i \leq M_t$); as channel size distribution is unknown by nodes, the adversary must therefore choose a valid channel to spoof at random.

Algorithm 9: Node replies to query, protocol #3

```
Description: As Alg. 7, except chain validation information is also
             generated and included in the reply payload.
1  $\mathcal{R} \leftarrow \emptyset$
   [...]                                              // as in alg. 7
30     $R$[height] $\leftarrow (0, \text{hash}(0|W|E|G_{i,1}))$          // chain info
31     OM(serialize($R$))
```

Algorithm 10: Node receives reply, protocol #3

```
Description: As Alg. 8, except chain validation is performed on
             responses prior to forwarding. Responses which fail this
             additional validation are dropped from the network.
1  $\mathcal{D} \leftarrow M$[data]
   [...]                                              // as in alg. 8
19             $K = M$[height]                      // chain validation
20             $S = K_1|K_1 - 1|\dots|1|0$                  // ordered seq.
21             if $K_1 \leq M_d \wedge hash(K_2|B.U) ==$
                 $hash(S|W|E|N.U|G_{i,2})$ then
22                 $M$[height] $\leftarrow (K_1 + 1, \text{hash}(K_1 + 1|K_2))$
23                 retransmit($M$)
```

D. Protocol #3: Query + Channel + Chain Validation

We extend the previous concept further, to leverage information about the network topology (if known) to help constrain and shape query response propagation. As discussed in footnote 2, multicasting may result in replies being routed more than needed. Although each node cannot track its own shortest route, the ledger can upper-bound a channel's response propagation through the application of *chain validation*.

1) Response Processing: Reply creation (Algorithm 9) is modified with the added inclusion of a new key. The first member of this newly-created 2-tuple serves as our current height, while the latter serves as the certification of that height.

Additional validation is added to response processing (Algorithm 10) to validate propagation height. Note that for chain validation, although M_d could be incremented by the adversary to truncate propagation, failing to forward the message at all would be more a more effective attack. Decrementation is disallowed due to the one-way nature of hashing.

2) Discussion: By leveraging information about network topology, and multicast-based reply propagation, nodes are able to determine a reply's degree of propagation. This is used to prune it from the network if it another identical copy would have already arrived at a bridge node through another route, without knowing that route's details.

We make use of the described hash-based construction to prevent adversaries from resetting the Height to a shorter distance, improperly extending message propagation. This ensures that although adversaries may prematurely remove messages from the network, they cannot boost the network's power consumption by promoting over-propagation of valid replies. A simple numeric counter, for example, would not provide such a guarantee.

One additional consideration is topological changes triggered by node captures or failures; such modifications may affect the longest-path bounds of a given node. This can be mitigated by increasing the M_d variable to include some flexibility for such changes; larger increases provide more resilience, at the cost of slightly higher transmission redundancy.

V. Protocol Assessment

A. Configuration & Setup

Topologies: Various deployment topologies were examined. Star describes a central node surrounded by neighbors. Tree describes hierarchical, balanced trees with branching factor 4. Line describes a linear sequence of connected nodes. Smallworld networks [23] are generated using the NetworkX v2.1 library [24] with $k = 4, p = 0.2$, where k is the number of linked nearest neighbors, and p is the likelihood of mutation. These parameters produce connected, ring-like networks with "shortcut" chordal edges. Powerlaw (or scale-free) networks [25] are similarly generated, with $m = 1, p = 0.5$ where m represents edges added incident to the existing graph, and p the likelihood of adding a triad. These values balance tightly-clustered graphs against excessive connectivity. 4-way Grids represent a lattice structure of nodes connected to their vertically- and horizontally-adjacent neighbors. 8-way grids are similar, with the addition of diagonally-adjacent edges.

Network sizes and dimensions were chosen to reflect both likely real-world scenarios as well as topologically-extreme possibilities, highlighting the different behaviors of the protocols under various conditions. Smallworld and powerlaw network generation parameters were chosen to best emulate ad hoc deployments, tuned to equalize node deployment density over the monitored area.

Topic Channel Membership: For topic channel evaluation, 100 channels were created to reflect complex rule-based access control policies, with varying counts of member nodes drawn randomly from the network, and permitting multi-channel memberships. Replies were evaluated with channels of 1%, 10%, and 40% node membership, chosen to reflect high, moderate, and low heterogeneity networks.

Attack Model: Adversarial evaluations assumed a 1% compromise rate of deployed nodes, and a 100% attack rate; that is, all compromised nodes attack simultaneously. This compromise rate balances adversarial resources against the risk of partition, a challenge not addressed in this work, while this attack rate reflects a worst-case scenario for our protocols that most-effectively highlights their strengths and weaknesses.

Environment: Evaluations were conducted using a network simulator implemented in Python3, running on an Intel i5-2500S@2.7GHz with 12GB RAM. Node deployments, channel assignments, and topological details were generated randomly, with results averaged over 25 iterations. Network size was set to $n = 1600$; similar patterns of results appeared for other sizes, but are omitted for brevity.

	Protocol 1	**Protocol 2**	**Protocol 3**
Star	2.558E+04	2.098E+04	1.600E+01
Tree		3.743E+03	8.651E+02
Line		1.536E+02	1.536E+02
Smallworld		2.007E+04	1.700E+04
Powerlaw		8.528E+03	8.286E+03
4-way Grid		1.902E+04	1.896E+04
8-way Grid		2.066E+04	2.057E+04

TABLE I: Transmissions Per Attack

	Protocol 1	**Protocol 2**	**Protocol 3**
Star	9.557E+02	4.148E+05	2.652E+03
Tree	9.593E+02	4.084E+05	2.871E+05
Line	9.506E+02	3.985E+05	2.813E+08
Smallworld	9.631E+02	4.214E+05	4.496E+06
Powerlaw	9.623E+02	4.125E+05	4.044E+06
4-way Grid	9.680E+02	4.256E+05	1.485E+07
8-way Grid	9.544E+02	4.144E+05	1.041E+07

TABLE II: Hash Operation Count per Query

B. Characteristics

Message Secrecy refers to the ability of nodes in the network, including adversarial nodes, to observe the contents of messages as they pass over the network. Since all three proposed protocols use hashing, they must also include their data payloads in the clear, as hash validation is strictly one-way. As such, their contents cannot be secret from an eavesdropper, and this approach is unsuitable for applications where the data should remain secret.

Capture Resilience describes the situation when a node in the network may become compromised by an attacker. Since our proposed protocols store only node-specific information on each device, the adversary may only sign messages using the local credentials of that captured node. Thus, if an adversary has control over 10% of the total nodes, they may only control at most 10% of a query's results. The baseline encryption approaches vary, with the shared-key approach failing totally after node capture, and the unique-key model proving equally secure to the proposed hash-based protocols.

C. Evaluation

Exhaustion Resilience refers to the ability of a protocol to prevent exhaustion attacks after a node capture; this would include actions like broadcasting duplicate or false messages designed to deplete power resources. In protocol #1, no additional steps are taken to prevent exhaustion attacks, but reduced hashing also results in lower computational costs. In protocol #2, propagation of falsified messages is curtailed, reducing adversarial impact and improving network durability. Finally, in protocol #3, these approaches are combined with chain validation to provide the most exhaustion resilience by upper-bounding the impact any given message, good or bad, can have on the network. The baseline encryption protocols provide an all-or-nothing solution, with results either fully-validatable (shared key) or not validatable at all (unique key). As seen in Table I, for networks with higher average betweenness centrality such as Lines, where the number of routes a message may take between nodes is fewer, the channel validation procedures described in protocol #2 provide reductions in adversarial

	Protocol 1	**Protocol 2**	**Protocol 3**	**Protocol 1**	**Protocol 2**	**Protocol 3**	**Protocol 1**	**Protocol 2**	**Protocol 3**
	1% Channel Membership Rate			**10% Channel Membership Rate**			**40% Channel Membership Rate**		
Star	2.597E+04		1.624E+01	2.523E+05		3.496E+02	1.025E+06		1.344E+03
Tree	2.504E+04		2.088E+03	2.570E+05		2.283E+04	1.018E+06		8.953E+04
Line	2.523E+04			2.564E+05			1.026E+06		
Smallworld	2.504E+04		2.501E+04	2.605E+05		2.601E+05	1.023E+06		1.022E+06
Powerlaw	2.638E+04		2.337E+04	2.578E+05		2.281E+05	1.026E+06		9.111E+05
4-way Grid	2.373E+04			2.523E+05			1.017E+06		
8-way Grid	2.699E+04			2.584E+05			1.028E+06		

TABLE III: Transmissions Per Query

propagation relative to protocol #1. As nodes associated with the queried channels will drop falsified messages, routes which include such nodes are pruned from the possible routing space. Using chain validation, Protocol #3 narrows these route choices further, increasing resilience by forcing falsified messages to use fewer, more-efficient routes.

Computational Efficiency: All three protocols take advantage of the reduced computational complexity of homomorphic hashing primitives relative to more-expensive encryption-based techniques. As seen in Table II, protocol #1 uses the fewest hash operations, while protocol #2 increases the rate proportional to the topic channel size – fewer, larger channels require more hashing. Protocol #3 varies based on the topology of the network, with increases during validation balancing against reductions from route pruning.

Network Efficiency refers to the ability to more effectively utilize network resources to accomplish query submission and response phases, as measured by the number of message transmissions. Query delivery is very efficient in all protocols. During the query response phase, added efficiency is offered by protocol #2 when adversaries are present, and by protocol #3 under certain deployment topologies. Topic channels provide substantial power savings by reducing the number of nodes which must respond to a single query – as seen in Table III, messaging complexity is a function of topic channel membership size. In cases where node heterogeneity is high and channel size is low, such savings can greatly extend the effective lifespan of the deployment without negatively affecting functional performance. As the maximal height of the chain validation is defined by the longest path to a bridge node from any other node, propagation sequences can be more strongly limited when these distances are shorter. Especially for topologies with small diameters, such as Star, protocol #3 provides the largest reductions in transmission costs due to more effective route pruning. Since protocol #3 also includes the behavior of previous validations, such reductions come in addition to those offered by channel validation.

VI. Conclusion

In this paper, we introduced on-demand topic channels in ad hoc ephemeral sensor deployments, and presented three alternative authentication protocols to provide secure, reliable, authenticated communication while providing resilience from adversarial or accidental disruption. Potential future work includes expanding on protocol behavior under partitioning, characterizing adversarial ability to partition under various topologies, and extending this paper's entrance node paradigm to include the concept of wear leveling to ensure that nodes within this set are not prematurely exhausted due to higher-than-usual communication loads. Additionally, intelligent message aggregation and forwarding provides several promising alternatives for further reducing network load.

References

[1] S.-H. Yang, "Wireless sensor networks: Principles, design and applications," *Springer*, 2014.

[2] S. D. Shirkande and R. A. Vatti, "Aco-based routing algorithms for ad-hoc network (wsn, manets): A survey," in *Communication Systems and Network Technologies (CSNT)*, 2013.

[3] N. Maalel *et al.*, "Reliability for emergency applications in internet of things," in *Distributed Computing in Sensor Systems (DCOSS)*, 2013.

[4] T. Qiu *et al.*, "Ergid: An efficient routing protocol for emergency response internet of things," *J. of Network and Computer App.*, 2016.

[5] D. Reina *et al.*, "A survey on multihop ad hoc networks for disaster response scenarios," *Intl. Journal of Distributed Sensor Networks*, 2015.

[6] J. Douglas *et al.*, "An open distributed architecture for sensor networks for risk management," *Sensors*, vol. 8, no. 3, pp. 1755–1773, 2008.

[7] M. T. Lazarescu, "Design of a wsn platform for long-term environmental monitoring for iot applications," *Journal on Emerging and Selected Topics in Circuits and Systems*, 2013.

[8] C.-S. Shih *et al.*, "Framework designs to enhance reliable and timely services of disaster management systems," in *Computer-Aided Design (ICCAD)*, 2016.

[9] U. Hunkeler *et al.*, "Mqtt-s - a publish/subscribe protocol for wireless sensor networks," in *COMSWARE*, 2008.

[10] M. N. Krohn *et al.*, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Security and Privacy*, 2004.

[11] J. Maitin-Shepard *et al.*, "Elliptic curve multiset hash," *The Computer Journal*, 2016.

[12] G. C. Pereira *et al.*, "Performance evaluation of cryptographic algorithms over iot platforms and operating systems," *Security and Comm. Networks*, 2017.

[13] O. Delgado-Mohatar *et al.*, "A light-weight authentication scheme for wireless sensor networks," *Ad Hoc Networks*, 2011.

[14] Z. Li and G. Gong, "Data aggregation integrity based on homomorphic primitives in sensor networks." in *ADHOC-NOW*. Springer, 2010.

[15] Q. Li and G. Cao, "Multicast authentication in the smart grid with one-time signature," *IEEE Transactions on Smart Grid*, 2011.

[16] Y. Hu *et al.*, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wireless networks*, 2005.

[17] A. Shaikh *et al.*, "Slim-simple lightweight and intuitive multicast protocol for manets," *International Journal of Computer Applications*, 2014.

[18] C.-H. Feng *et al.*, "Stateless multicast protocol for ad hoc networks," *Trans. on Mobile Computing*, 2012.

[19] P. J. Leach *et al.*, "A universally unique identifier (uuid) urn namespace," 2005.

[20] A. Perrig *et al.*, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[21] L. Junhai and et.al., "A survey of multicast routing protocols for mobile ad-hoc networks," *Comm. Surveys & Tutorials*, 2009.

[22] NIST, "Sp 800-154: Guide to data-centric system threat modeling," 2016.

[23] T. G. Lewis, "Small-world networks," *Network Science: Theory and Practice*, pp. 131–175, 2009.

[24] A. Hagberg *et al.*, "Exploring network structure, dynamics, and function using networkx," *Los Alamos Nat. Lab*, 2008.

[25] A.-L. Barabási, "Scale-free networks: a decade and beyond," *science*, vol. 325, no. 5939, pp. 412–413, 2009.