Low-Complexity Secure Protocols to Defend Cyber-Physical Systems Against Network Isolation Attacks

Dong-Hoon Shin*, Jinkyu Koo†, Lei Yang*, Xiaojun Lin†, Saurabh Bagchi†, and Junshan Zhang*
*School of Electrical, Computer and Energy Engineering, Arizona State University, USA
†School of Electrical and Computer Engineering, Purdue University, USA
Email: donghoon.shin.2@asu.edu, kooj@purdue.edu, lyang55@asu.edu,
{linx, sbagchi}@purdue.edu, junshan.zhang@asu.edu

***Abstract*—This paper studies the *network isolation attack*, a devastating type of attacks on cyber-physical systems. In this attack, an adversary compromises a set of nodes that enclose a region in order to isolate the region from the rest of the network. Assuming that the compromised nodes wish not to be detected, we propose a solution to defend against the network isolation attack. Our goal is to achieve the following security guarantee: either a legitimate node can successfully deliver a message to another legitimate node, or the network control center can identify a small set of suspect nodes, which are guaranteed to contain a compromised node. Toward achieving this goal, we develop two protocols: one is for secure delivery of messages among nodes and the other is for secure collection of messages from nodes at the network control center. We show that our proposed protocols are provably secure, i.e., attain the aforementioned security guarantee. Further, our protocols achieve this guarantee with overhead that is *orders-of-magnitude smaller* than existing baseline protocols. Our proposed protocols are thus scalable for large networks.**

I. Introduction

With the growth of networked computing technologies, the capabilities of computation and communication are being deeply embedded in physical systems. Such a tight integration of physical systems and advanced computing technologies is leading to a new generation of engineered systems, called *Cyber-Physical Systems* (CPS). A CPS harnesses the new capabilities of computation and communication to control and manage the physical systems, thereby providing highly dependable, efficient and performance-enhanced systems. CPS can potentially benefit various applications and areas, including electric grid, health care, transportation and military.

Although CPS can benefit greatly from the use of communication and networking technologies, they also become increasingly dependent upon the communication network for the operation of the physical system. CPS thus give rise to additional security vulnerabilities. Moreover, in many cases, CPS serve as the critical infrastructures to the public, such as electricity, water, oil and gas. This makes the communication networks for CPS a major target for adversaries (e.g., terrorists) who intend to cause severe damage to a large population.

One type of attacks that can cause a devastating damage to CPS is *network isolation attack* or simply *isolation attack*, where a set of compromised nodes would result in the isolation of a region from the rest of the network. The isolation attack can be viewed as a form of coordinated black-hole (or packet-dropping) attacks. In the attacks, an adversary compromises a set of nodes that enclose a region. It can then disconnect the nodes in the region from those outside the region by dropping all packets coming into, or going out of, the region. Here, the compromised nodes could behave more intelligently by selectively dropping packets or mixing with other malicious activities such as modifying and delaying packets. Further, if the adversary is powerful, the compromised nodes can collude with each other to launch a stronger form of attacks, e.g., wormhole attack. In general, it would be very difficult to protect the network against this kind of massive attacks. Once such an isolation attack is launched, it can cause serious damage to the system, which can lead to a critical system failure when the isolated region is large. Moreover, an adversary can incur serious damage to network with a (relatively) small cost, i.e., by targeting a small number of topologically-critical nodes (i.e., hub nodes) to launch the isolation attack.[1]

Defending networks against malicious activities has been studied extensively in the literature. Nevertheless, the existing works have paid little attention to the isolation attack, perhaps because it is an "extreme" form of attack, in the sense that once it is launched, there are few ways to protect the network against compromised nodes. In fact, one can argue that there is no solution that can ensure communication among (legitimate) nodes under the isolation attack. This is because multi-hop communications among nodes must go through other possibly compromised nodes, and these nodes can arbitrarily deny the service and not follow any protocol. Indeed, once an adversary completely encloses an isolated region, no communication would be possible between a node within the isolated region and a node outside the isolated region.

In view of the severe damage caused by potential isolation attacks, we would have liked to develop a protocol to achieve the following ideal guarantee: no matter what forms of iso-

[1]In general, half the nodes in the network could be disconnected from the other half by $O(\sqrt{n})$ number of compromised nodes, where n is the total number of nodes in the network.

lation attacks are launched, our protocol can always ensure communication between legitimate nodes. Unfortunately, this guarantee is impossible to achieve as aforementioned. To make progress, we need to impose some additional restrictions on the adversary. Indeed, such restrictions should be as mild as possible so that it is applicable to cover a large number of scenarios. In this paper, we impose such a restriction of a "perfect crime" on the adversary, i.e., the compromised nodes wish not to be detected by the defensive measures of the network. This scenario is of great interest because once detected, the compromised nodes can be removed by technicians (or soldiers in battle-fields) dispatched by the central authority (or the command center). Hence, assuming it is not always an easy task to compromise a legitimate node (e.g., through the use of reasonable security mechanisms such as anti-tamper hardware), the adversary would have liked not to be detected when engaging in malicious actions. This implies that, assuming the perfect-crime restriction on the adversary, if we can always identify a subset of compromised nodes whenever they behave adversarially, we can then force them to follow the correct behavior.

Under this perfect-crime assumption, our objective in this paper is to build a system that can ensure communications between legitimate nodes when there are no malicious activities; however, if the adversary ever misbehaves and violates the protocol, the system would identify a small set of suspect nodes that must contain at least one compromised node. Once we build such a system, the adversary must either follow the network protocol, or expose one of its compromised nodes to the system administrator, who can then remove it. We emphasize that even this weaker notion of security guarantee could be difficult to attain. As compromised nodes can collude to launch stronger attacks, one may not be able to identify the culprit, especially under the setting when a majority of nodes in a local neighborhood are malicious [1].

Further, a primary goal here is to not only build such a system that accomplishes the above security objective under the isolation attack, but also minimize the complexity of the system so that the system is scalable for large networks. Toward achieving this goal, we develop the following two protocols: *Low-complexity Secure Delivery Protocol* (LSDP) and *Low-complexity Secure Collection Protocol* (LSCP). LSDP allows a legitimate source to *securely deliver* a message to a legitimate destination, while LSCP allows a legitimate source to *securely collect* (and deliver) messages from intermediate nodes on a path (to a legitimate destination). Here, by "secure delivery" and "secure collection", we mean that they can achieve the aforementioned security guarantee. We show that the overhead of the two protocols is *orders-of-magnitude smaller* than that of a straightforward approach—which extends the underlying schemes in existing works in a straightforward manner. *To the best of our knowledge, this paper is the first attempt to provide such security guarantees under the isolation attack, under general network settings*[2].

The rest of the paper is organized as follows. Section II introduces the network and the attack model. Section III describes a formal definition of our design objective and the research challenge in this paper. Sections IV and V present the two basic protocols that we develop—LSDP and LSCP—and prove their security guarantees. Section VI analyzes the overhead of the proposed protocols. Section VII presents performance evaluation of the proposed protocols. Section VIII discusses prior works related to this paper. Finally, Section IX gives conclusions and discusses future works.

II. Network and Attack Model

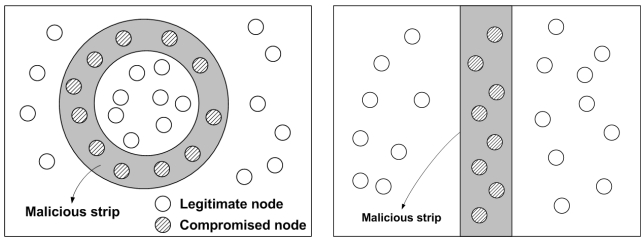
A. Network Model

We consider a stationary network, where nodes do not move and are connected via wireless or wired links. A set of nodes in the network, called *collecting stations* (CSs), are deployed to collect emergency messages or reports from nodes. We state a few assumptions on the security capabilities of nodes and the setting that we are interested in. Nodes are legitimate when they are deployed, but they may be compromised as time goes by (as commonly assumed in the literature). For example, nodes can be compromised during software update, or can be physically tampered if they are deployed in insecure locations (e.g., mesh routers deployed on rooftops or street-lights). CSs are trustworthy, i.e., are protected from attacks by an adversary. There exists a Certification Authority (CA) that administrates a public-key infrastructure. Every node thus knows the public key of every other node. Further, the private key of a legitimate (i.e., uncompromised) node is only known to itself. Each pair of neighboring nodes has a shared secret key (established during their neighbor discovery phase). Thus, two neighboring nodes are capable of generating message authentication codes (MACs) to authenticate and provide integrity on messages between them.

B. Considered Attack

We consider the Byzantine adversary model, i.e., compromised nodes can perform arbitrary malicious activities, such as (selectively) dropping, modifying and delaying packets, and even wormhole attacks. Under this adversarial model, we focus on an attack launched by a set of compromised nodes to isolate a region from the rest of the network. We refer to this attack as the *network isolation attack* (or simply, *isolation attack*). Figure 1 illustrates two forms of isolation attack in wireless networks. The first form (in Fig. 1(a)), called *ring isolation attack*, is launched by enclosing a region by a strip filled with compromised nodes. The second form (in Fig. 1(b)), called *bar*

[2]Our prior work [2] has previously provided a similar security guarantee to that of this paper. However, the work [2] focuses on a different problem: timely and secure delivery of event reports to a base station in wireless sensor networks. On the other hand, this paper targets more general application settings, i.e., any-to-any communication, and focuses on reducing the overhead targeting for large networks. Besides, this paper has a very different solution approach from [2]: This paper presents *on-demand* protocols whereas [2] presents a *proactive* protocol. (Refer to Section VIII for detailed discussion.)



(a) Ring isolation attack. (b) Bar isolation attack.

Fig. 1. Isolation attacks.

isolation, is launched more efficiently than the ring isolation by exploiting the network boundary.

In wireless networks, an adversary can also isolate a region by launching a coordinated jamming attack. However, for the ease of exposition, we assume that the physical layer uses jamming-resilient schemes, such as spread-spectrum techniques (as in the 802.11), to defend against jamming attacks.

III. Design Objective and Research Challenge

In this section, we first define our design objective and then discuss the main research challenges that this paper addresses.

A. Design Objective

We first define some terminologies.

***Definition** 1:* A message is said to be *correctly delivered* if the message is delivered to the destination within a time bound and without being modified. Also, a message is said to be *securely delivered* to the destination if either the message is correctly delivered, or the source can narrow down to a set of at most two *suspect* nodes, where at least one of the suspect nodes must be a compromised node.

Here, the compromised node would be either the culprit node that disrupted the message delivery, or another compromised node that colluded with the culprit node. We call the set of suspect nodes the *suspect set*. If the suspect set contains a single node, then the node must be compromised. On the other hand, when the suspect set contains two nodes, one of them may be legitimate. In this case, the two suspect nodes usually have a disagreement, implicitly accusing each other of lying. One may wish to identify who is the compromised node from the suspect set by employing a more complex agreement scheme. However, this is provably impossible under certain circumstances, e.g., when a majority of nodes in a local neighborhood are malicious [1].

In addition, we will regard the inability to communicate between two nearby legitimate nodes caused by a natural node (or link) failures (despite multiple retransmissions) also as a malicious activity. This is reasonable because it is often impossible to distinguish with absolute certainty between faulty nodes (due to natural causes) and malicious nodes. Thus, unless stated otherwise, a suspect set must contain either at least one compromised node or at least one faulty node.

The design objective is defined as follows.

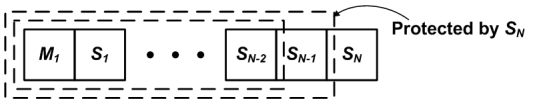


Fig. 2. A form of expensive secure acknowledgement. It contains the signatures created in an onion manner by intermediate nodes $n_1, \ldots, n_N$ on a path. M is a signed message sent by the source, and S_i is the onion-manner signature signed by the i-th intermediate node n_i on the path.

***Definition** 2 (Design objective):* The design objective is to ensure *secure delivery* of messages between two legitimate nodes, and also to let CSs be informed of the suspect nodes identified whenever messages are not correctly delivered, no matter what forms of isolation attacks are launched.

B. Challenge

One could develop a solution to accomplish the above design objective by extending the secure acknowledgement schemes presented in the existing works [3], [4]. Specifically, in this solution, the source requires a confirmation from the destination through a known path. This confirmation is delivered to the source in a form of secure acknowledgement, which requires signatures from all intermediate nodes in addition to the destination. The signatures of the intermediate nodes are created in an "onion" manner (see Fig. 2). That is, each intermediate node signs on the entire message of the destination's confirmation and all signatures accumulated so far. When an intermediate node does not receive a secure acknowledgement within a bounded time, the node gives up waiting and generates its own acknowledgement. This acknowledgement is delivered to the source in the form of the secure acknowledgement. Due to the onion-manner signatures, the source can detect any deletion or modification of any part of the secure acknowledgement, and also localize a faulty/malicious node.

However, such a straightforward extension of the existing works to attain the secure delivery (i.e., the first part of the design objective) would be too costly due to the excessive use of the expensive onion-manner signatures, and it would not be scalable for large networks. Further, this type of protocols may still have difficulty providing the secure guarantee in Def. 2. Specifically, it is conceivable that such protocols may ensure that, when the message delivery is unsuccessful, the *source node* can find a suspect set. However, this detection capability may be guaranteed only at the source node. Hence, if the source nodes are within an isolated region including no CS, they will have no way to deliver the detected suspect sets to a CS. A trivial remedy so that CSs can collect the detected suspect sets would be to probe all nodes individually for detection results. However, if not designed carefully, such a naïve approach would be very costly. Hence, it is still unclear how CSs (thus the system administrator) can efficiently collect detected suspect sets from nodes.

In summary, there remain two major challenges in finding a *low-complexity* solution that attains the design objective (Def. 2):

Algorithm 1 Procedure for the source $\mathcal{S}$

Sends an MSG to N_1
Sets a timer that will go off after time t_s, and waits for an ACK/NACK from N_1
if receives an ACK/NACK within t_s **then**
  Verifies the followings:
    i) MAC for the ACK/NACK (generated by N_1)
    ii) $\mathcal{D}$'s signature (in the ACK) or the accuser's signature (in the NACK)
  if finds an incorrect MAC/signature **then**
    Terminates and concludes that N_1 is compromised
  else
    if it is an ACK **then**
      Terminates and concludes that the delivery was successful
    else
      Let N_s and N_a be the suspect node and the accuser, respectively, reported in the NACK
      Terminates and concludes that N_s and N_a are suspect
    end if
  end if
else
  Terminates and concludes that N_1 is compromised
end if

Algorithm 2 Procedure for intermediate nodes N_i for $i \in \{1, \ldots, n-1\}$ and destination $\mathcal{D} = N_n$

// Denote a suspect node by SN and $N_0 = \mathcal{S}$
Waits for an MSG from N_{i-1}
if receives an MSG **then**
  Verifies the followings:
    i) MAC for the MSG (generated by N_{i-1})
    ii) $\mathcal{S}$'s signature in the MSG
  if finds an incorrect MAC/signature **then**
    Generates a NACK with SN $= N_{i-1}$, then sends the NACK to N_{i-1} // terminates
  else
    if N_i is not the destination **then**
      Replaces the N_{i-1}'s MAC by N_i's MAC, then sends the MSG with N_i's MAC to N_{i+1}
      Sets a timer that will go off after time t_i, then waits for an ACK/NACK from N_{i+1}
      if receives an ACK/NACK within t_i **then**
        Verifies the followings:
          i) MAC for ACK/NACK (generated by N_{i+1})
          ii) $\mathcal{D}$'s signature (in the ACK) or the accuser's signature (in the NACK)
        if finds an incorrect MAC/signature **then**
          Generates a NACK with SN $= N_{i+1}$, then sends the NACK to N_{i-1} // terminates
        else
          Replaces the N_{i+1}'s MAC by N_i's MAC, then sends the ACK/NACK with N_i's MAC to N_{i-1} // terminates
        end if
      else
        Generates a NACK with SN $= N_{i+1}$, then sends the NACK to N_{i-1} // terminates
      end if
    else
      Generates an ACK, then sends the ACK to N_n // terminates
    end if
  end if
end if

1) How can we design a *low-complexity* protocol that can guarantee the secure delivery of messages between two legitimate nodes?
2) How can CSs collect the detected suspect sets in a *cost-effective* manner?

In the following two sections, we develop two protocols, called LSDP and LSCP, to address these two challenges respectively. LSDP is used by nodes for secure delivery of messages among them, and LSCP is employed by CSs to securely collect detected suspect sets from nodes so that they can be removed or reprogramed. LSDP and LSCP can reduce the overhead by an order of magnitude and half an order of magnitude, respectively, compared to the straightforward approach (see Section VI).

IV. Low-complexity Secure Delivery Protocol

In this section, we present the *Low-complexity Secure Delivery Protocol*, termed LSDP. It ensures that a legitimate source $\mathcal{S}$ securely delivers a message to a legitimate destination $\mathcal{D}$ through a given path. The basic idea of LSDP is to let each node take the responsibility of securely delivering the source's message from itself to the destination. To achieve this, each node makes use of a time-bounded secure acknowledgement (ACK) and a negative-acknowledgement (NACK). The key difference from the straightforward approach described in Section III-B is that LSDP requires *at most one* MAC in any packet, except the source's signature signed on the message (see Fig. 3). As a result, LSDP reduces the overhead significantly, i.e., by an order-of-magnitude smaller than the straightforward approach (see Section VI).

The basic procedure of LSDP is described in Alg. 1 for the source node and Alg. 2 for the intermediate nodes and the destination. In the procedure, we assume that the path consists of nodes $\mathcal{S}$, $N_1, \ldots, N_{n-1}$, $\mathcal{D}$. There are three types of messages exchanged among nodes, which contain the following information:

- MSG: 1) $\mathcal{S}$'s message; 2) $\mathcal{S}$'s signature signed on the $\mathcal{S}$'s message with the $\mathcal{S}$'s private key
- ACK: 1) $\mathcal{D}$'s confirmation; 2) $\mathcal{D}$'s signature signed on the $\mathcal{D}$'s confirmation with the $\mathcal{D}$'s private key
- NACK: 1) a suspect node; 2) the accuser's ID (i.e., the

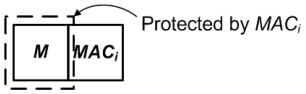


Fig. 3. The format of any packet sent at an intermediate node N_i in LSDP. It contains *only one* MAC (except the source's signature on the message M).

ID of the node that accuses the suspect node); 3) the accuser's signature signed on the concatenation of the suspect node and the accuser's ID with the accuser's private key.

Each message is sent with the MAC generated by the sender, only when the sender is not the originator of the message. The MAC is for the authentication of the message. Note that since a MAC for a message is generated by the secret key shared only between the sender and the receiver. If both the nodes are legitimate, the MAC cannot be generated by any other nodes. By using the MACs (generated using the same symmetric key), instead of digital signatures (generated using asymmetric key), we make the protocol computationally light. The digital signatures may still be used before a pairwise shared key is established between the source and each intermediate node. This pairwise shared key between the two nodes can be established on demand by the source, which generates and sends a key by encrypting it with its private key.

The time-out values, i.e., t_s for $\mathcal{S}$'s timer (in Alg. 1) and t_n for N_i's timer (in Alg. 2), can be set by taking into account the number of nodes in the path and the time that each node needs to successfully send a message following Alg. 2 (allowing multiple retransmissions for unreliable wireless links). For example, suppose that the number of the intermediate nodes in the path is $n-1$ and T is a sufficient time for each node to successfully send a message following Alg. 2. We can set the time-outs as $t_s = (2n-1)T$ and $t_i = (2(n-i)-1)T$.

LSDP attains the following security guarantee.

***Theorem** 1:* Through LSDP, a legitimate node can *securely deliver* a message to another legitimate node through a known path.

Proof: After $\mathcal{S}$ sends an MSG to N_1, there will be three possible cases; $\mathcal{S}$ receives 1) nothing; 2) ACK; 3) NACK.

Case 1: Receive nothing. In this case (line 15 in Alg. 1), $\mathcal{S}$ will conclude that N_1 is compromised. This conclusion is correct since otherwise, i.e., if N_1 is legitimate, N_1 should have sent either an ACK or a NACK to $\mathcal{S}$ following Alg. 2.

Case 2: Receives ACK. In this case, there are two possibilities: i) the ACK contains an incorrect MAC/signature; ii) both the MAC and the signature are correct. In the first case (line 6 in Alg. 1), if the MAC is incorrect, obviously, N_1 must be compromised. Or, if the $\mathcal{D}$'s signature is incorrect, it must be true that the $\mathcal{D}$'s confirmation or signature has been modified by either N_1 or another intermediate node. Note that, in the latter case, if N_1 is legitimate, N_1 should have sent a NACK following Alg. 2. Hence, for the both cases, N_1 must be compromised. In the second case (line 9 in Alg. 1), i.e., if both the MAC and the $\mathcal{D}$'s signature in the ACK are correct, clearly, the MSG must have been correctly delivered to $\mathcal{D}$. Thus, for both of the possibilities in this Case 2, $\mathcal{S}$ will either correctly deliver the MSG or obtain a correct suspect set.

Case 3: Receives NACK. In this case, there are two possibilities: i) the NACK contains an incorrect MAC/signature; ii) both the MAC and the signature are correct. In the first case (line 6 in Alg. 1), N_1 must be compromised for the same reason as in the case when the ACK contains an incorrect MAC/signature (in the Case 2). In the second case (line 11 in Alg. 1), we will show that $\{N_s, N_a\}$ forms a correct suspect set. If the accuser N_a is compromised, then by definition, $\{N_s, N_a\}$ forms a correct suspect set. Hence, we only need to consider the other case where N_a is legitimate and to show that N_s is compromised. Since the N_a's signature in NACK is correct, it must be N_a who has generated the NACK received by $\mathcal{S}$. Also, according to Alg. 2, the reason why N_a generated the NACK must be either because N_a had found an incorrect MAC/signature in the ACK/NACK from N_s (lines 6 and 14 in Alg. 2), or because N_a had received nothing from N_s until its timer expired (line 19 in Alg. 2). Obviously, none of these two cases would have happened if N_s is legitimate (recall that no link failure is assumed). This means that N_s must be compromised, and hence $\{N_s, N_a\}$ forms a correct suspect set. Thus, for the both possibilities in this Case 3, $\mathcal{S}$ obtains a correct suspect set.

Therefore, for all of the Cases 1, 2 and 3, it is true that $\mathcal{S}$ will either correctly deliver the MSG to $\mathcal{D}$, or obtain a correct suspect set. Thus, the theorem follows. ■

This theorem means that given a path, any legitimate node can securely deliver a message to another legitimate node through Alg. 1 and Alg. 2. These paths can be given by the network administrator (since nodes are stationary), or can be found through any of the existing routing protocols (e.g., OSPF) when nodes were initially deployed. In fact, to make the message delivery even more reliable, each source can compute multiple disjoint paths to the destination. Our protocol ensures that for each path, a suspect set will be identified if the message is not delivered correctly. Thus, if one path fails, the source can try another path that does not contain the detected suspect nodes.

V. Low-complexity Secure Collection Protocol

In this section, we address the second challenge stated in Section III-B. Our solution approach is to have CSs collect detected suspect sets from nodes. A trivial solution would be to let CSs probe nodes one by one by employing LSDP. Then, in the returned ACK, each node can report the suspect sets that it has identified. However, such a trivial solution would be costly because probing all nodes individually creates a large number of messages in the network. This solution would not only make CSs and their neighbors overloaded, but also incur high overhead to the entire network, i.e., high expense on bandwidth, energy and computational resources. Ideally, we would like to have a more efficient solution that can probe a large number of nodes in one round. To this end, we develop the *Low-complexity Secure Collection Protocol*, termed LSCP. The basic idea of LSCP is to divide a path of the nodes to

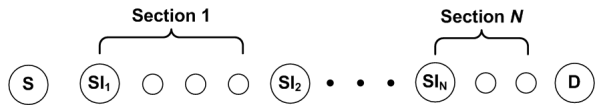


Fig. 4. LSCP divides a path into multiple sections, and each section has a section inspector (SI) that takes a responsibility to gather the detected suspect nodes in its section and include them into the CT.

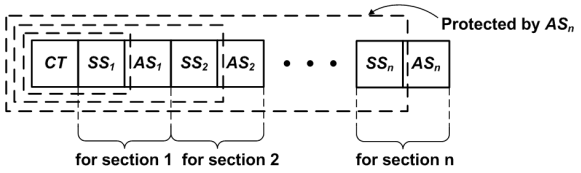


Fig. 5. The format of CT sent at the end of section n in LSCP. It has a clear distinction from the expensive secure acknowledgement (in Fig. 2): it contains *only one* aggregated signature for the detected suspect sets from *each section*, i.e., all nodes in the section. SS_i and AS_i denote the detected suspect sets collected in section i and the aggregated signature for section i that is signed on top of S_{i-1} aggregately by the all nodes in the section, respectively.

be probed into multiple sections and appoint a node in each section as a *section inspector* (SI) (see Fig. 4). Each SI gathers detected suspect sets from nodes in its section and includes them into a *collection token* (CT) sent by a CS.

However, such a simple use of the SIs alone would not achieve the goal of the end-to-end secure collection, since SIs may be compromised. Note that a compromised SI may not include certain detected suspect sets into the CT. Hence, our main challenge in designing LSCP is how to develop a *low-complexity* protocol to ensure the end-to-end secure collection along a path of *potentially compromised* SIs, which can also *collude*. The key idea to achieve this goal is to use a single "aggregated signature" for a verifiable proof by which the CS can confirm that every node in each section received and verified the CT. Here, the node verifies the CT to check whether the SI in its section has indeed included into the CT all of its detected suspect sets that the SI had gathered.

The aggregated signature is constructed as follows. Suppose that there are nodes $N_1, \cdots, N_I$ in a section. Let k_i^{pr} and k_i^{pu} be the N_i's private key and public key, respectively. Denote the encryption and the decryption of a message M with a key k by $E(M,k)$ and $D(M,k)$, respectively. Also, denote the aggregated signature of N_i on S_M by SGN_i, where S_M is an SI's signature signed on a message M. We create SGN_i as the following: $\text{SGN}_j = E(\text{SGN}_{j-1}, k_j^{pr})$ for $j \in \{1, \ldots, i\}$, where $\text{SGN}_0 = S_M$. Then, the SI in the next section can verify the correctness of SGN_I by the following steps: the SI first decrypts it sequentially with the nodes' public keys, i.e., $\text{SGN}_{i-1} = D(\text{SGN}_i, k_i^{pu})$ for $i \in \{I, \ldots, 1\}$. At the end of this process, the SI verifies the correctness of SGN_0, i.e., whether SGN_0 is indeed the SI's signature signed on M. Note that the SI needs only M and a *single* aggregated signature to verify that all nodes in its section have signed on M.

The procedure of LSCP is described in Alg. 3, 4 and 5. Here, the source, i.e., a CS, sends a CT through a path of the nodes that the source wants to probe for detected suspect sets. A CT collects the detected suspect sets from the nodes on the path, and delivers them to the destination, i.e,. another CS. Depending on whether the source needs to collect the detected suspect sets or not, which is an option for the network operator using LSCP, the CT may or may not be piggybacked on the ACK to return to the source. If the CT returns to the source (CS), for which Algs. 3, 4 and 5 are presented, the source verifies the signatures in the CT and identifies a suspect set if needed. If it does not, the destination (CS) instead performs these verification and identification procedures. The path is divided into multiple sections such that the numbers of nodes in the sections are as equal as possible. An SI is also a node, but has the following extra duties: (a) to gather the detected suspect sets from the nodes in its section; (b) to include them into the CT; and (c) to sign on the entire CT using its signature.

Algorithm 3 Procedure for the source $\mathcal{S}$

```
Sends a CT to SI_1
Sets a timer that will go off after time t_s, and waits for an ACK/NACK from SI_1
if receives an ACK/NACK within t_s then
    Verifies the followings:
      i) MAC for the ACK/NACK (generated by SI_1)
     ii) D's signature (in the ACK) or the accuser's signature (in the NACK)
    iii) Aggregated signatures in the CT, which are verified in the reverse order of the path
    if it is an ACK then
        if MAC and all signatures in the ACK are correct then
            Terminates and concludes that the CT is correct
        else if MAC or D's signature is incorrect then
            Terminates and concludes that SI_1 is compromised
        else
            // i.e., if finds an incorrect aggregated signature
            Let the section n be the first section where an incorrect signature is found
            Terminates and concludes that SI_{n+1} is compromised
        end if
    else
        // i.e., if it is a NACK
        if finds an incorrect MAC/signature in the NACK then
            Terminates and concludes that SI_1 is compromised
        else
            Let X_s and X_a be the suspect node and the accuser, respectively, reported in the NACK
            Terminates and concludes that X_s and X_a are suspect
        end if
    end if
else
    Terminates and concludes that SI_1 is compromised
end if
```

Algorithm 4 Procedure for node N_i

1: // Assume that N_i is in section n (where SI_n is in charge)
2: // SN and SS denote a suspect node and a suspect set
3: Waits for a CT from its previous node P (= N_{i-1} or SI_n)
4: **if** receives a CT **then**
5: Verifies the followings in the CT:
i) P's aggregated signature
ii) Whether the SSs sent to SI_n are all included
6: **if** the P's aggregated signature is incorrect **then**
7: Generates a NACK with SN $= P$, then sends the NACK to P // terminates
8: **else if** some of the SSs sent to SI_n are missing **then**
9: Generates a NACK with SN $= \mathrm{SI}_n$, then sends the NACK to P // terminates
10: **else**
11: Generates an aggregated signature by signing on the P's aggregated signature with its private key
12: Replaces the P's aggregated signature in the CT with the new one
13: Sends the CT to the next node, then performs the procedure in lines 10–20 in Alg. 2 // terminates
14: **end if**
15: **end if**

Algorithm 5 Procedure for section inspector SI_n

1: // SN and SS denote a suspect node and a suspect set
2: // If $\mathrm{SI}_n = \mathrm{SI}_1$, then N_i and N_i's aggregated signature in the procedure below are replaced by $\mathcal{S}$ and $\mathcal{S}$'s signature
3: Wait for a CT from the previous node N_i
4: **if** receives a CT **then**
5: Verifies the N_i's aggregated signature in the CT
6: **if** the N_i's aggregated signature is incorrect **then**
7: Generates a NACK with SN $= N_i$, then sends the NACK to N_i // terminates
8: **else**
9: **if** SI_n is not the destination **then**
10: Attaches to the CT the followings:
i) N_i's aggregated signature
ii) the SSs sent from the nodes in its section and the SSs detected by itself
11: Signs on the entire CT, including the signatures and SSs accumulated so far, with its private key
12: Sends the signed CT to the next node, then follows the procedure in lines 10–20 in Alg. 2 // terminates
13: **else**
14: Attaches the N_i's aggregated signature and the SSs detected by itself to the CT, and signs on the entire CT with its private key
15: Generates an ACK of the signed CT, then sends the ACK to N_i // terminates
16: **end if**
17: **end if**
18: **end if**

The CT initially contains only a signed request message (like an MSG). Thereafter, at each SI, the CT gains two additional attachments: the suspect sets detected in the section and the aggregated signature for the section (see Fig. 5). Here, the suspect sets are, in fact, the NACKs generated by the accusers, and each suspect set consists of the suspect node and the accuser included in a NACK. Thus, a false accuser is also contained in the suspect set.

The local collection of detected suspect sets at each SI can be done by the SI in the following manner: upon receiving a CT, it individually probes the nodes in its section using LSDP. Or, to facilitate the collection process, each node in a section uses LSDP to securely deliver the suspect sets whenever it detects. However, both of these potential procedures have an issue: in the former, a compromised SI may not probe certain nodes in its section; in the latter, a compromised node may prevent a legitimate node from delivering the detected suspect sets to the SI. Note that, in the latter case, the legitimate node would find a suspect set (due to the guarantee of LSDP), but still this suspect set is known to only that node, not to a CS. To resolve this issue, nodes perform the following additional procedure: if a node either receives a CT without being probed by its SI (in the former case), or finds an unsuccessful delivery of the detected suspect sets to its SI (in the latter case), then upon receiving a CT, the node drops it and sends a NACK containing the detected suspect node toward the source, i.e., a CS. (Here, the detected suspect node would be the compromised SI or the newly found suspect node.) Then, due to the guarantee of LSCP (Theorem 2, which we will show later), either the NACK will be correctly delivered to the source (CS), or the CS will find a suspect set. In this way, we can ensure that either each node can correctly deliver its detected suspect sets to its SI, or a CS can find a suspect set.

Note that the key idea of using the aggregated signatures is embedded in Alg. 4 (in line 5). The first verification is to check whether SI_n has included all of the detected suspect sets reported by N_i into the CT. Second, the verification of the aggregated signature is to prevent a possible wormhole attack launched by two (or more) colluding compromised SIs. To elaborate this, suppose that SI_n and SI_{n+1} are compromised and collude with each other to launch the following attack: upon receiving a CT, SI_n includes none of the suspect sets detected in its section into the CT, and then sends the CT directly to SI_{n+1} over an out-of-band channel (in wireless networks) to bypass all the nodes in the section. However, such a wormhole attack can be detected by the source (CS) that verifies the aggregated signatures in the CT. Specifically, the source verifies the aggregated signatures in the CT in the reverse order of the path. On verification, the source would find an incorrect aggregated signature in the section n, since all nodes in this section are bypassed and thus none of their signatures are aggregated. Thereafter, following Alg. 5, the CS would conclude that SI_{n+1} is compromised.

The LSCP achieves the following security guarantee.

***Theorem** 2:* Through LSCP, a legitimate node can securely collect (and deliver) messages from the intermediate nodes on a known path (to another legitimate node).

Proof: After $\mathcal{S}$ sends a CT, there will be three possible cases; $\mathcal{S}$ receives 1) nothing; 2) ACK; 3) NACK.

Case 1: Receive nothing. In this case (line 22 in Alg. 3), $\mathcal{S}$ will obtain a correct suspect set, as shown in the Case 1 in the proof of Theorem 1.

Case 2: Receives ACK. In this case (line 5 in Alg. 3), there are three possibilities: i) the SI_1's MAC and all the signatures in the ACK are correct; ii) the SI_1's MAC or the $\mathcal{D}$'s signature in the ACK is incorrect; iii) an aggregated signature in the CT is incorrect. In the first case (line 7 in Alg. 3), clearly, $\mathcal{S}$ will obtain a correct CT. In the second case (line 9 in Alg. 3), SI_1 must be compromised, since otherwise, i.e., if SI_1 is legitimate, SI_1 should have sent a NACK to $\mathcal{S}$ according to Alg. 4. In the third case (line 11 in Alg. 3), if we assume that SI_{n+1} is legitimate, then upon receiving the CT, SI_{n+1} would have found an incorrect signature in the CT and then have generated a NACK. However, SI_{n+1} has sent the ACK violating the protocol. Therefore, SI_{n+1} must be compromised. Thus, for all of the three possibilities in this Case 2, $\mathcal{S}$ will either obtain a correct CT or a correct suspect set.

Case 3: Receives NACK. In this case (line 14 in Alg. 3), there are two possibilities: i) the NACK contains an incorrect signature; ii) both MAC and signature in the NACK are correct. In the first case (line 16 in Alg. 3), if SI_1 is legitimate, SI_1 should have sent a NACK containing all correct signatures, according to Alg. 4. However, this did not happen, and therefore SI_1 must be compromised. We now consider the second case (line 18 in Alg. 3). We show that $\{X_s, X_a\}$ forms a correct suspect set. If the accuser X_a is compromised, then by definition, $\{X_s, X_a\}$ forms a correct suspect set. Hence, we only need to consider the other case, where X_a is legitimate, and to show that X_s is compromised. Since all the signatures in NACK are correct, it must be X_a that has generated the NACK received by $\mathcal{S}$. Also, since X_a is legitimate, X_a would have generated the NACK due to one of the following events being occurred: lines 7, 9 and 13—lines 14 and 19 in Alg. 2—in Alg. 4, and lines 7 and 12—lines 14 and 19 in Alg. 2—in Alg. 5. It is easy to verify that none of these events would have happened if X_s is legitimate (recall that no link failure is assumed). This means that X_s must be compromised, and consequently $\{X_s, X_a\}$ forms a correct suspect set. Thus, for the both possibilities in this Case 3, $\mathcal{S}$ will obtain a correct suspect set.

Therefore, for all the three cases, $\mathcal{S}$ will either obtain a correct CT or a correct suspect set. Thus, the theorem follows. ■

VI. Overhead Analysis

In this section, we first evaluate the overhead of LSDP and LSCP, and then compare with the overhead of the straightforward approach (described in Section III-B). We measure the overhead of the protocols in terms of the total number of cryptographic check values (i.e., signatures and MACs) being transmitted throughout the protocol execution on a given path. We use this metric since it is the major factor in the increase of the payload size in the protocols. Thus, the metric also determines how much extra network resources, i.e., the bandwidth, energy and computational resources, are expended due to the use of the protocol.

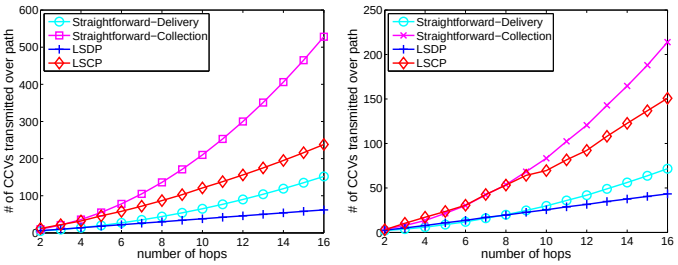
Overhead of LSDP. In LSDP, all types of the messages exchanged between nodes contain only one signature. Therefore, any packet, which may or may not include a MAC, will contain at most two cryptographic check values. Thus, the total number of cryptographic check values transmitted is $O(l)$, where l is the number of nodes in the given path. Note that this is the minimum level of cryptographic check values transmitted for any protocol: for the security objective under consideration, any packet at each node has to include at least two cryptographic check values—one for the authentication of the source's message and the other for the authentication of the immediate sender of the packet.

Overhead of LSCP. For ease of exposition, suppose that the given path consists of $l = h^2$ intermediate nodes (except for the source and the destination). We divide the path into h sections, each of which has h nodes including an SI. As a CT travels across each section, the number of cryptographic check values included in the CT increases only by one, due to the aggregated signature for the section. Therefore, the size of CT (thus ACK) grows linearly with h. Thus, any packet contains at most $O(h)$ cryptographic check values. It is easy to verify that, at each section, a total of $O(h^2)$ cryptographic check values are transmitted to perform the local collection of detected suspect sets through LSDP. Since the path consists of h sections, the number of the cryptographic check values transmitted to perform this location collection for all sections is $O(h^3)$, i.e., $O(l\sqrt{l})$. Thus, the total number of the cryptographic check values transmitted is $O(l\sqrt{l})$.

We now compare the overhead of LSDP and LSCP with that of the straightforward approach. In the straightforward approach applied to the secure delivery and the secure collection, the numbers of signatures to be attached in a packet would grow linearly with the number of nodes in the path, i.e., l, for both protocols. Hence, the total numbers of the cryptographic check values transmitted are both $O(l^2)$. On the other hand, as shown above, the overhead of LSDP and LSCP are $O(l)$ and $O(l\sqrt{l})$, respectively. Thus, LSDP and LSCP reduce the overhead by an order of magnitude and half an order of magnitude, respectively, compared to the straightforward approach.

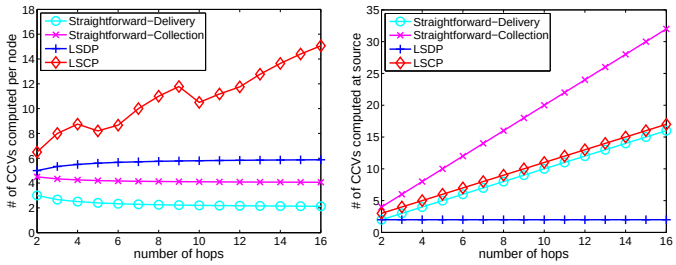
VII. Numerical Results

In this section, we evaluate our protocols using simulations. We compare the communication overhead and the computational overhead of our protocols to those of a straightforward extension of the existing schemes [3], [4] (described in Section III-B). These two types of overhead are measured by the total number of cryptographic check values (CCVs) transmitted over a given path and the average number of CCVs computed per node, respectively. We simulate the protocols



(a) Normal scenario. (b) Attack scenario.

Fig. 6. Communication overhead.



(a) Per node. (b) At source node.

Fig. 7. Computational overhead (for normal scenario).

on one path of various lengths to see how the overhead grows with the path length. For LSCP, we determine the number of sections on a path by rounding the square root of the number of hops (to make the numbers of nodes in the sections as equal as possible). We consider two scenarios: normal and attack scenarios. In the normal scenario, there is no malicious node and thus the ACK will be correctly delivered to the source. In the attack scenario, one of the intermediate nodes on the path is compromised, and its position is randomly chosen with equal probability to every node. The compromised node drops either a message/CT toward the destination, or an ACK toward the source. It randomly performs either of the two malicious behaviors with equal probability. For this attack scenario, we show the average overhead taken over 1000 times of the simulation.

Figure 6 shows the communication overhead of the protocols for the two scenarios. In both figures, we observe that our protocols—LSDP and LSCP—reduce the overhead significantly. Further, this overhead reduction becomes larger as the path length increases, as expected from the overhead analysis in Section VI. Note the large overhead gap between the straightforward extensions for the secure delivery and for the secure collection. This is because the straightforward extension for the secure collection has to attach one onion-manner signature to CT at each intermediate node, and also the returned ACK has to accompany with the CT including all the signatures attached.

Figure 7 shows the computational overhead of the protocols for normal scenario. Fig. 7(a) shows the average number of CCVs computed per node. We observe that LSCP has increasing computational overhead (where the abrupt decreases are due to the number of sections being changed), while the others have constant computational overhead. This is because LSCP requires each node to decrypt the aggregated signature received from its previous node, and the computational overhead for this decryption increases linearly with the number of nodes in the section. We can view this increasing computational overhead of LSCP as the cost that LSCP pays for the reduction of communication overhead. Fig. 7(b) shows the number of CCVs computed at the source. We observe that LSDP and LSCP require the source to compute less number of CCVs, compared to those using the straightforward approach. From this and the result in Fig. 7(a), we can see that our protocols lower the computational load at the source by making the load distributed over the nodes on the path. This is desirable, especially for the networks that have a small number of CSs (due to the high cost for their strong protection against powerful adversaries). In LSCP, the high computation load at the CSs will be distributed over all nodes in the network.

VIII. Related Work

There have been a large volume of works on defending networks against malicious activities, especially in ad hoc wireless networks. Nevertheless, most of the existing works have paid little attention to the isolation attack, and hence have little or limited effectiveness against the isolation attack. Conventional secure routing protocols (e.g. [5]–[7]) provide little help under the isolation attack. Most of the secure routing protocols are based on the assumption that there exists a secure route between the source and the destination, i.e., a route that consists entirely of legitimate nodes. However, no such path would exist under the isolation attack. Path-quality monitoring techniques (e.g., [8]–[11]) can raise an alarm when misbehaviors by intermediate nodes on the path are detected. However, they only detect misbehaviors on the path, without identifying the malicious nodes. Overhearing-based approaches (e.g., [12]–[17]), where nodes watch for their neighbors by overhearing the neighbors' communication (exploiting the omni-propagation nature of wireless signals), are effective against the basic black-hole attack or a weak form of isolation attacks. That is, they are effective only when the width of the *malicious strip*, which is the strip of the region filled with compromised nodes (see Fig. 1 in Section II), is thin. However, the overhearing-based approaches are ineffective to the stronger form of isolation attacks launched by multiple consecutive and colluding malicious nodes. Specifically, in this stronger form of isolation attacks, the first malicious node forwards a packet and then the subsequent malicious node drops it. But, the first malicious node intentionally hides (i.e., does not report) this fact, thereby prevents any legitimate node from identifying which node has dropped the packet.

Recent works [2]–[4], including our own prior work, have studied detecting and identifying malicious activities in the network. However, they have different focuses and approaches. The works [3], [4] have proposed faulty-link localization schemes against Byzantine [3] or packet-dropping [4] adver-

saries. These schemes use a form of secure acknowledgement that requires *onion-manner* signatures (see Fig. 2) to identify a faulty link. As we discussed in Section III-B, one could extend these schemes to develop a defense mechanism against the isolation attack. However, this would be very costly due to the excessive use of the expensive onion-manner signatures (refer to Section III-B for the detailed discussion). Further, the works [3], [4] did not address our second challenge, i.e., the cost-effective mechanism for CSs to collect detected suspect sets from nodes. Our prior work [2] has proposed an adversary identification protocol against Byzantine adversarial nodes. It provides a security guarantee that those attained by our protocols are similar to. However, the work [2] focuses on a different problem: timely and secure delivery of event reports to a base station in wireless sensor networks. On the other hand, our work targets more general application settings, i.e., any-to-any communication, and focuses on reducing the overhead for large networks. Also, our solution approach is very different from that of [2]. Our protocols are *on demand*, i.e., are used only when nodes need to send a packet or collect messages from nodes, whereas the work [2] presents a proactive protocol that periodically sends a packet to collect event reports from nodes in a timely and secure manner.

IX. Conclusion

In this paper, we study defense mechanisms against the network isolation attack to cyber-physical systems (CPS), in which compromised nodes would isolate a region from the rest of the network. The impact of this type of attack can be devastating: without proper defense mechanisms, the adversary can isolate a large region by compromising a (relatively) small number of nodes that enclose the region. Assuming that the compromised nodes wish not to be detected, we develop a solution to defend against the isolation attack. Our solution achieves the following provable security guarantee: either a legitimate node can successfully deliver a message to another legitimate node, or the network control center can identify a pair of suspect nodes, which is guaranteed to contain at least one compromised node. A key contribution of our proposed solution is to achieve this guarantee with the overhead that is orders-of-magnitude smaller than existing baseline protocols. Thus, our solution is scalable for large networks.

For future work, we plan to study the deployment issues, including how to place the collecting stations and form the collecting paths in the network, and the cost-performance trade-offs of the proposed defense mechanisms.

References

[1] L. Lamport, R. Shostak, and M. Peasee, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, July 1982.

[2] J. Koo, D.-H. Shin, X. Lin, and S. Bagchi, "A Delay-Bounded Event-Monitoring and Adversary-Identification Protocol in Resource-Constraint Sensor Networks," *Elsevier Ad Hoc Networks*, vol. 11, no. 6, pp. 1820–1835, August 2013.

[3] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "ODSBR: An On-Demand Secure Byzantine Resilient Routing Protocol for Wireless Ad Hoc Networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 4, pp. 1–35, January 2008.

[4] X. Zhang, A. Jain, and A. Perrig, "Packet-dropping Adversary Identification for Data Plane Security," in *ACM CoNEXT*, 2008.

[5] P. Papadimitratos and Z. Haas, "Secure Routing for Mobile Ad Hoc Networks," in *SCS CNDS*, 2002.

[6] Y.-C. Hu, D. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," in *IEEE WMCSA*, 2002.

[7] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 21–38, January 2005.

[8] N. G. Duffield and M. Grossglauser, "Trajectory Sampling for Direct Traffic Observation," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 3, pp. 280–292, June 2001.

[9] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Improving Accuracy in End-to-End Packet Loss Measurement," in *ACM SIGCOMM*, 2005.

[10] J. Sommers and N. Duffield, "Accurate and Efficient SLA Compliance Monitoring," in *ACM SIGCOMM*, 2007.

[11] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford, "Path-Quality Monitoring in the Presence of Adversaries," in *ACM SIGMETRICS*, 2008.

[12] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," in *ACM MobiCom*, 2000.

[13] I. Khalil, S. Bagchi, and N. Shroff, "LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks," in *IEEE/IFIP DSN*, 2005.

[14] D.-H. Shin and S. Bagchi, "Optimal Monitoring in Multi-Channel Multi-Radio Wireless Mesh Networks," in *ACM MobiHoc*, 2009.

[15] D.-H. Shin, S. Bagchi, and C.-C. Wang, "Distributed Online Channel Assignment Toward Optimal Monitoring in Multi-Channel Wireless Networks," in *IEEE INFOCOM, Mini-conference*, 2012.

[16] D.-H. Shin and S. Bagchi, "An Optimization Framework for Monitoring Multi-Channel Multi-Radio Wireless Mesh Networks," *Elsevier Ad Hoc Networks*, vol. 11, no. 3, pp. 926–943, May 2013.

[17] D.-H. Shin, S. Bagchi, and C.-C. Wang, "Toward Optimal Sniffer-Channel Assignment for Reliable Monitoring in Multi-Channel Wireless Networks," in *IEEE SECON*, 2013.