

## Neutralization of Errors and Attacks in Wireless Ad Hoc Networks

Claudio Basile, Zbigniew Kalbarczyk, Ravi K. Iyer  
Center for Reliable and High-Performance Computing  
University of Illinois at Urbana-Champaign, IL 61801  
{basilecl, kalbar, iyer}@crhc.uiuc.edu

### Abstract

*This paper proposes and evaluates strategies to build reliable and secure wireless ad hoc networks. Our contribution is based on the notion of inner-circle consistency, where local node interaction is used to neutralize errors/attacks at the source, both preventing errors/attacks from propagating in the network and improving the fidelity of the propagated information. We achieve this goal by combining statistical (a proposed fault-tolerant cluster algorithm) and security (threshold cryptography) techniques with application-aware checks to exploit the data/computation that is partially and naturally replicated in wireless applications. We have prototyped an inner-circle framework with the ns-2 network simulator, and we use it to demonstrate the idea of inner-circle consistency in two significant wireless scenarios: (1) the neutralization of black hole attacks in AODV networks and (2) the neutralization of sensor errors in a target detection/localization application executed over a wireless sensor network.*

### 1. Introduction

A wireless ad hoc network is a group of nodes that are capable of forming a network without any pre-fixed infrastructure. Wireless ad hoc networks (ranging from mobile networks of laptops/PDAs to sensor networks) are highly unstable, highly susceptible to accidental errors (in software and hardware components), and easy targets of security attacks. Importantly, these problems stem from the very nature of wireless networks, i.e., node mobility, deployment in harsh environments, need for low-cost solutions, limited availability of communication/computation/energy resources, and broadcast communication [1]. The goal of this paper is to propose and evaluate strategies to build wireless ad hoc networks that continue to operate correctly in hostile computing environments, even if some of the nodes have been compromised by errors<sup>1</sup> or attacks. We make the following contributions:

- Introduction of the notion of *inner-circle consistency*, where local node interaction (in a one-hop neighborhood) is used to neutralize errors/attacks at the source,

<sup>1</sup>The focus of this work is not on transmission errors (e.g., due to fading) but on errors in the computation, communication, or sensing units of wireless ad hoc nodes. Causes of these errors include hardware transients, software bugs, and device degradation.

both preventing errors/attacks from propagating in the network and improving the fidelity of the propagated information. We achieve this goal by combining (1) a *secure topology* service, which discovers the local network topology; (2) a *deterministic voting* technique, which embeds application-aware checks that validate the information disseminated by exploiting the data/computation that is partially and naturally replicated in wireless applications (e.g., neighboring AODV [2] nodes may compute similar routing information, while near sensor nodes may collect similar environmental data); (3) a *statistical voting* technique, which improves the accuracy of the propagated information and removes hidden error/attack data by means of a proposed *fault-tolerant cluster* algorithm; and (4) threshold cryptography, which guarantees message integrity despite node intrusions.

- Design and formal specification of an *inner-circle framework* for wireless ad hoc nodes, a reconfigurable architecture that provides a common substrate in which one can embed a wide range of error/attack-neutralization techniques. By trading off a targeted dependability level (i.e., guarantees on message integrity) with resource usage, the framework can be scaled to the communication, computation, and energy resources available on a wireless node. The architecture spans both software modules and hardware modules (*Crypto-Processor* for tamper-resistant key-store plus signature creation/verification, *Fault-Tolerant Cluster Processor* for faulty/malicious data masking).
- Prototype and evaluation of the inner-circle framework with the ns-2 network simulator [3] using two significant wireless scenarios: (1) the neutralization of black hole attacks in AODV networks and (2) the neutralization of sensor errors in a target detection/localization application run over a wireless sensor network. The first scenario is motivated by the devastating impact of black hole attacks (e.g., 3000% throughput reduction in our experiments) and by the absence of an effective countermeasure up to the time of this writing [4, 5]. We show that the inner-circle approach can limit the throughput degradation to below 22%. The second scenario is motivated by the experimental evidence that wireless sensor networks can be very unreliable due to sensor devices, which interact directly with the environment and

fail days before the electronics may fail [6]. We show that the inner-circle approach can halve energy consumption (double network lifetime) while providing a four-to-six-fold improvement in target detection latency and target localization accuracy.

## 2. System Model

The system considered comprises a set  $\mathcal{N}$  of mobile nodes that communicate by exchanging messages through wireless channels. A node does *not* know the complete set  $\mathcal{N}$  but can discover nodes in its proximity by means of periodic beacons. Correct nodes are associated with unique ids that they maintain throughout their life and are aware of their geographic position [7]. A threshold cryptography scheme is available [8].

A *dependability level* is an integer  $L \geq 1$  that a (source) node  $x$  specifies when it wants to diffuse a piece of information  $I$  in the network. Example criteria for the choice of  $L$  include the importance of the information and the size of the node’s neighborhood. The inner-circle approach permits a (remote) recipient node  $y$  of information  $I$  to infer whether  $I$  was agreed upon by  $L$  neighbors of source  $x$ . To support this mechanism, we limit dependability level  $L$  to vary within a predetermined range (e.g.,  $1 \leq L \leq 10$ ) and associate a secret signing key  $K_L$  with each value of  $L$ .  $K_L$  is not disclosed to any node but each node only has an  $(L + 1)$ -threshold share of  $K_L$  (thus,  $L + 1$  nodes must cooperate to sign a message with secret key  $K_L$ ). For simplicity, we assume that wireless nodes obtain their signing key shares from a trusted dealer at the system’s initialization time; moreover, we do not discuss potential extensions to support proactive secret sharing [9].

The system computation is modeled by extending the *timed asynchronous* model [10] to wireless ad hoc networks. Each node has access to a local hardware clock, but node clocks are not required to be globally synchronized.

**Node Failure.** Nodes may fail by *crashing* or by becoming *Byzantine*. The cause for a node failure may be an accidental error (e.g., a transient in the hardware or a software bug) or an adversary that has compromised the node (e.g., by stealing the node’s secret keys or by reprogramming its software to execute malicious code). *Correct* nodes never fail.

**Timely Connectivity.** In the absence of failures and node movement, each node is stably connected to its neighbors, and wireless communication channels are *timely*, i.e., if a node  $p$  sends a message to a neighbor  $q$  at a time  $t$ , then  $q$  receives the message by time  $t + \delta$  (where  $\delta$  includes both transmission and processing delays). In that case we say that  $p$  is *t-connected* with  $q$  at time  $t$ . Nonetheless, the system executions we consider are subject to *communication failures*, which can be temporary (e.g., a single untimely reception due to collisions) or permanent (e.g., no further reception due to node movement or adversary jamming).

**Adversary.** We make the following assumptions about an adversary: (1) The cryptographic primitives used by correct nodes are secure. (2) The adversary has limited jamming

range and cannot disrupt communication in the whole network. (3) Compromised nodes are not all capable of colluding by communicating (e.g., through out-of-band channels) and sharing their identities/secret keys [11]. When nodes do collude, we count them for the number of different identities they can present.

## 3. Overview of Inner-circle Consistency

This section introduces the idea of inner-circle consistency through an example. Subsequent sections provide a more detailed presentation of the inner-circle consistency algorithms.

**Execution Scenario.** Consider a hierarchical wireless sensor network deployed over a remote region with the task of determining the presence of targets/events of interest (e.g., a fire in a forest) [12]. At the lowest level, a large number of *sensor nodes*, constrained in energy and computation resources, gather environmental data (e.g., sound, temperature, pressure samples); then, a middle layer of more powerful *collector nodes* aggregates the data from the sensor nodes and forwards them to an upper layer of *access points*, which constitute the gateways to the external world. The remainder of this example focuses on the middle-layer nodes.

Suppose that a collector node  $x$  receives data from a group of sensor nodes in its proximity, computes an aggregate value  $v$  (e.g., presence/absence of a target) based on the received data, and propagates value  $v$  to a distant access point. A sensed signal can be affected by environmental noise; thus, value  $v$  can suffer from a natural accuracy error. Also, node  $x$  may be faulty (e.g., physically damaged due to high humidity or high temperature) or malicious (e.g., compromised by an adversary) and may report invalid or inconsistent values (e.g., wrong target type, or target detection to some access points and no target to the other access points); this can disrupt the upper-level node coordination necessary to accomplish additional tasks (e.g., tracking the detected target).

**Inner-circle Concept.** To neutralize errors/attacks originating from a collector node  $x$ , all the collector nodes that are (securely discovered) neighbors of  $x$  form an *inner-circle* that checks and filters any value originating from  $x$ . To send a value  $v$ , node  $x$  selects a *dependability level*  $L$  (see § 2) and initiates an inner-circle voting protocol. The protocol involves only  $x$ ’s inner-circle nodes and terminates only if at least  $L$  neighbors agree on value  $v$ —we assume that collector nodes are deployed densely enough that if a collector node  $x$  detects a target, with high probability,  $L$  of its neighbors also do so.

**Inner-circle Mechanism.** In agreeing on a proposed value, the inner-circle voting protocol enables (1) application-aware checking (*deterministic voting*) by enforcing that an agreed value  $v$  satisfies application-specific criteria (e.g.,  $v$  is contained within a predetermined range) and (2) improving the accuracy of the proposed value (*statistical voting*) by combining it with values from  $x$ ’s inner-circle nodes.

Whereas  $x$ ’s errors are handled by its inner-circle nodes through a deterministic/statistical voting protocol, one must

also consider possible corruption of the value(s) forwarded outside the inner-circle due to faulty/malicious inner-circle nodes and faulty/malicious nodes on the forwarding paths. In our approach, the messages an inner-circle propagates in the network (as a result of the execution of a voting protocol) are self-checking, i.e., a recipient node  $y$  outside  $x$ 's inner-circle can check whether a message  $m$  it receives from node  $x$  includes a value agreed upon by at least  $L$  of  $x$ 's inner-circle nodes, where  $L$  is the dependability level indicated by message  $m$ . This is achieved through threshold signatures [8].

When initiating an inner-circle voting protocol, a node  $x$  sends a propose message that specifies the dependability level  $L$  that it wishes to use. An inner-circle node that agrees with  $x$ 's proposed value acknowledges its agreement by replying with a partial signature obtained with its share of secret  $K_L$ . If at least  $L$  neighbors reply to  $x$  (i.e., the inner-circle voting protocol terminates) then  $x$  fuses the received  $L$  partial signatures with its generated partial signature and obtains a signature  $\sigma_{K_L}$  of the agreed value  $v$  with secret  $K_L$ . At this point, node  $x$  encapsulates value  $v$  in an *agreed* message  $m$  that includes value  $v$ , dependability level  $L$ , and the combined signature  $\sigma_{K_L}$ . On receiving message  $m$ , a remote node  $y$  verifies the validity of the included signature  $\sigma_{K_L}$  (to infer whether the included  $v$  was agreed upon by  $L$  neighbors of  $x$ ) before delivering value  $v$  to the local application.

**Discussion.** A unique characteristic of the inner-circle approach is that expensive intrusion- and fault-tolerant algorithms are executed only in the proximity of a source node. The advantage is three-fold: (1) Local interaction enables fast detection and suppression of errors/attacks, which both prevents errors/attacks from escalating and avoids an unnecessary waste of resources, e.g., messages sent, energy consumed. (2) Executing complex protocols only locally helps reduce communication overhead and energy consumption; also, node mobility only affects the execution of the protocol instances run in a locality of the moving nodes. (3) Application-aware checking can be more efficient when performed locally, where redundant application information can be readily available. Conducting this check far from the source may be questionable due to node movement and transmission delay, which may obsolete the checked information.

The inner-circle approach does trade off performance for dependability, since the number of errors/attacks it can tolerate is limited by the size of the inner-circle, which can be less than what one can theoretically achieve with standard fault tolerance algorithms run across the entire wireless network [13]. Defining larger inner-circles (e.g., including all nodes two hops away from a source node) can effectively rebalance this trade-off; for ease of presentation, however, the remainder of this paper focuses on the case of one-hop inner-circles in a homogeneous set of nodes.

## 4. Inner-circle Consistency Node Architecture

This section introduces a wireless node architecture for implementing inner-circle consistency applications (see Fig. 1). At the bottom of the architecture, a *Physical Layer*, a *Medium Access Control (MAC) Layer*, and a *Link Layer* provide best-effort, single-hop unicast/multicast communication. These services are abstracted out as a *Single-hop Communication Service*. At the top of the architecture, the *Application* represents a user application that runs on the wireless node, and the *Routing and Forwarding Service* corresponds to the ad hoc routing and forwarding mechanisms implemented in the wireless node in support of multi-hop communication. These services are not specific to the inner-circle approach.

Five components are unique to the inner-circle architecture: (1) An *Inner-circle Interceptor*, which intercepts messages to/from the link layer and performs extra actions for those messages that match a registered message template (i.e., a description of the messages for which the application requests inner-circle checking). Matching outgoing messages are redirected to the inner-circle services, while matching incoming messages are suppressed if they have originated from a suspected node (i.e., a potentially misbehaving node) or if the messages' signatures are incorrect. (2) A *Suspicious Manager*, which receives node misbehavior indications from other inner-circle services and maintains a list of the suspected nodes. The mechanism is such that a node  $p$  suspects a node  $q$  permanently only if  $p$  has a provable evidence of  $q$ 's misbehavior (e.g., when  $p$  receives a message  $m$  that is properly signed by  $q$  but has an invalid field or violates the currently executing protocol); otherwise, suspicion is only temporary (e.g., for a few minutes). (3) A *Secure Topology Service*, which enables nodes to discover the topology of their neighborhood in a secure manner and to determine in which inner-circle they should participate. (4) An *Inner-circle Voting Service*, which enables nodes to perform deterministic or statistical voting—as specified by the application—on the messages/values sent by an inner-circle's center (or source) node. (5) A set of *Inner-circle Callbacks*, which are application-provided callback functions that supply an application-specific customization to the inner-circle voting service and are invoked in response to events occurring in the node, e.g., arrival of a message that needs to be checked.

The proposed inner-circle consistency node architecture can be customized depending on the available resources and the characteristics of the application and of the wireless environment. Figure 2(a) and Fig. 2(b) provide two instantiations of the architecture for ad hoc nodes and sensor nodes, respectively. In the examples, the physical layer is fully hardware-implemented, the MAC layer is partially firmware, and the link layer is an integral part of the operating system (kernel or micro-kernel). Routing and forwarding service is provided by AODV and Directed Diffusion protocols [2, 14]. The architecture also includes two dedicated hardware modules: a *Crypto-Processor*, which provides tamper-resistant key-store plus key

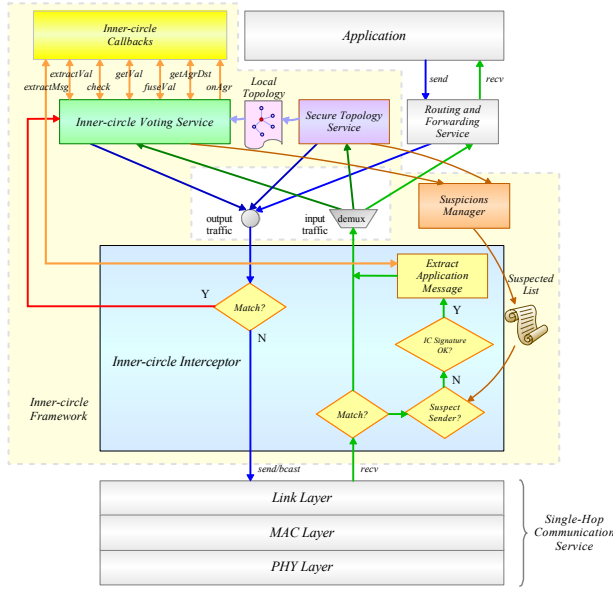


Figure 1. Inner-circle consistency node architecture.

cryptographic functions (i.e., signature generation and verification), and a *Fault-Tolerant Cluster Processor*, which provides key error/attack masking functions (i.e., fault-tolerant cluster algorithm of § 4.3). These modules can guarantee high protection against malicious tampering with the wireless nodes, high performance, and low energy consumption (up to two orders of magnitude less energy than in software implementations). An early Crypto-Processor was introduced in [15] and has been reconfigured to trade off performance for area occupation and energy consumption. A Fault-Tolerant Cluster Processor has also been developed.

The operating system (Linux for ad hoc nodes and TinyOS for sensor nodes) is augmented with the Inner-circle Interceptor. The interceptor is implemented in Linux as a loadable kernel module and in TinyOS as a TinyOS component that exports a send/receive TinyOS interface, which is directly used by the sensor application. Other inner-circle services (voting service, secure topology service, and suspicions manager) are implemented as user-level daemons in Linux and as TinyOS components in TinyOS. Applications access inner-circle services via an API that allows them (1) to initiate and configure the services (e.g., selection of deterministic versus statistical voting, selection of dependability level  $L$ ), (2) to specify *message templates* that describe the application messages to be checked (the architecture enables selective use of the inner-circle approach, as not all application messages are necessarily checked by the inner-circle services), and (3) to specify a set of Inner-circle Callbacks (whose code resides in a shared library in Linux and in a TinyOS component in TinyOS).

The following sections discuss the main inner-circle services in greater detail. Due to space limitations, formal service specifications are relegated to [16].

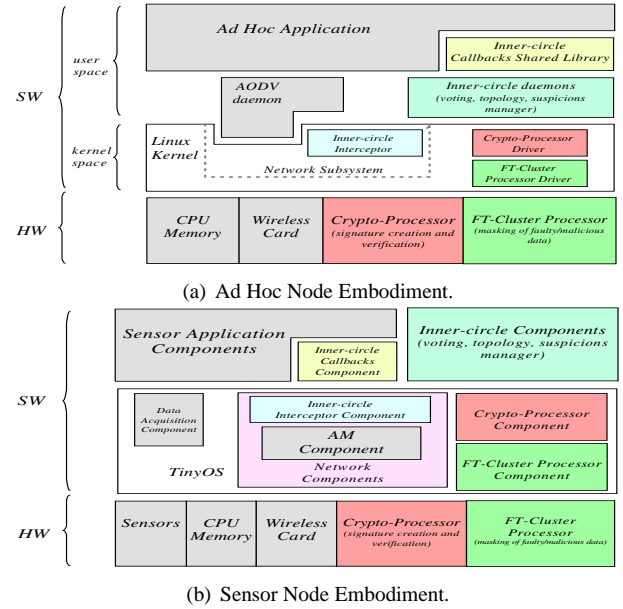


Figure 2. Inner-circle consistency embodiments.

#### 4.1 Secure Topology Service

A *Secure Topology Service (STS)* discovers and authenticates bidirectional links up to two hops away and provides each node with a local topology view. (Two hops are necessary, since a node needs to authenticate its neighbors' neighbors in order to securely participate in its neighbors' inner-circles.) Informally, we assume that there is a known time interval  $\Delta_{STS}$  such that an STS implementation satisfies a *Completeness* property, which formalizes the ability to exclude untimely (e.g., broken) links, and *One-Hop Accuracy* and *Two-Hop Accuracy* properties, which capture the ability to include one- and two-hop timely links. The proposed STS implementation assumes that local clocks at neighboring nodes are kept (approximately) synchronized, and it operates by periodic broadcasting of STS messages (with period  $\tau < \Delta_{STS}/2$ ). An STS message originating from a correct node  $p$  includes a list of  $p$ 's neighbors, authenticated through an extension of the fixed Needham-Schroeder protocol [17].

#### 4.2 Inner-circle Voting Service

An *Inner-circle Voting Service (IVS)* implements two voting schemes (see Fig. 3). *Deterministic voting* prevents illegitimate values from being propagated in the wireless network. A value  $v$  agreed upon by  $c$ 's inner-circle nodes is the value initially proposed by  $c$ , and it is agreed upon only if it complies with an application-dependent criterion  $f$  (the `check` method of the Inner-circle Callbacks in Fig. 1). *Statistical voting* improves a proposed value  $v_c$ 's accuracy. The value agreed upon by the inner-circle nodes is obtained by the statistical fusion of an original value  $v_c$  from  $c$  with corresponding values  $v_p$  from other inner-circle members  $p$  via a fault-tolerant fusion function  $f$  (the `fuseVal` method of the Inner-

circle Callbacks). Importantly, function  $f$  must cope with arbitrary values sent by faulty/malicious inner-circle members (see § 4.3). The objective of statistical voting is to enable reliable and secure *in-network processing*, where intermediate nodes aggregate data collected from a group of nodes before forwarding the aggregated data to the proper destinations.

Inner-circle voting algorithms are parameterized by a dependability level  $L$ , associated with each agreed message, that indicates the number of inner-circle nodes that must cooperate with a center (or source) node  $c$ . Based on the maximum number  $F$  of node failures to tolerate—where  $F_B$  are Byzantine,  $F_C$  are crashes, and  $F_L$  are due to broken links<sup>2</sup>—and the number  $N$  of nodes in an inner-circle (including the center node), node  $c$  sets dependability level  $L$  so as to guarantee a minimum number  $T$  of non-Byzantine participants in each IVS execution that completes successfully. It can be shown that setting  $L = N - F - 1$  guarantees *Agreement*, *Integrity*, and *Termination* properties of IVS protocols (introduced below) for  $T = L - F_B$ . As a special case, fixing  $L + 1 = 2N/3$  and ignoring  $F_C$  and  $F_L$  provides tolerance to  $N/3 - 1$  Byzantine failures and guarantees that a majority of correct nodes must agree for the protocol to terminate; this scenario corresponds to standard Byzantine agreement algorithms.

Informally, the *Agreement* property states that a sender node  $c$  needs approval from at least  $T$  non-Byzantine inner-circle nodes in order to assemble a valid agreed message  $m$  that indicates a dependability level  $L$ . The *Integrity* property states that a (remote) node  $y$  receiving an agreed message  $m$  that indicates a dependability level  $L$  can rely on the information contained in  $m$ . The *Termination* property states that an IVS algorithm execution initiated by a correct node  $c$  is guaranteed to terminate. Note that IVS algorithms are not required to terminate if the initiator node is faulty.

### 4.3. Fault-Tolerant Value Fusion

This section discusses techniques to implement the fault-tolerant fusion function  $f$  used in § 4.2 to support reliable and secure *in-network processing*. The mathematical problem considered is the (fault-tolerant) estimation of an unknown (vector) parameter  $\Theta$  from a set of  $L$  (vector) observations  $P = \{p_1, \dots, p_L\}$  that are corrupted by random noise such that  $p_i = \Theta + N_i$ , where  $N_i$  are i.i.d. zero-mean random variables. In contrast with classical estimation theory, we allow up to a number  $F$  of these observations to be arbitrarily corrupted (beyond noise  $N_i$ ), owing to faults/attacks.

A simple implementation of function  $f$  is given by a *fault-tolerant mean* algorithm originally proposed in the context of approximate agreement [18] and then applied to fault-tolerant in-network processing [19]. Fault-tolerant mean is just one of the many algorithms proposed for approximate agreement and

<sup>2</sup>If  $c$  is not Byzantine, then  $F_B$  accounts only for Byzantine nodes in  $c$ 's inner-circle; otherwise,  $F_B$  accounts also for Byzantine nodes that an agreed message may encounter during its traversal of the wireless network.  $F_C$  and  $F_L$  always account only for failures in  $c$ 's inner-circle.

clock synchronization [20, 21]. The limitation of these techniques, when applied to in-network processing, is that they always discard a number of input observations, which results in limited accuracy even in the common case of no faulty data. High accuracy is important for the inner-circle approach because local value fusion is done on the data collected by a limited number of nodes—an inner-circle is not expected to have more than 10–15 members [22].

**Fault-Tolerant Cluster Algorithm.** This paper contributes with the proposal of a *Fault-Tolerant Cluster* algorithm to generate, from a set  $P$  of  $L$  observations, an estimate  $\hat{\Theta}_{FT}$  that is highly accurate yet robust to faulty/malicious data (in  $P$ ). To achieve this goal, we exclude from the estimation process only those observations that are likely to be faulty/malicious, i.e., that are inconsistent with the distribution indicated by the remaining observations. Before presenting the algorithm, however, a few definitions are required.

Given a cluster (or set of data points)  $C$  and a point  $p \in C$ , we define the distance  $d(p, C)$  of point  $p$  from cluster  $C$ :

$$d(p, C) = \|p - \text{centroid}(C \setminus p)\| \quad (1)$$

where  $\text{centroid}(\{p_1, \dots, p_n\}) = \sum_{i=1}^n p_i/n$ . The justification for the above definition is that we want distance  $d(p, C)$  to be proportional to the information lost when excluding  $p$  from  $C$ . Indeed, consider a set  $C' = C \setminus p$  of i.i.d. observations having a unimodal p.d.f., symmetric w.r.t. the mean  $\mu$ . The farther point  $p$  is from  $\mu$ —for which  $\text{centroid}(C')$  is an estimator—the lower the probability of observing point  $p$ , and thus, the larger the information<sup>3</sup> carried by such an observation.

The *fault-tolerant cluster*  $C_p^*$  is defined as the maximum-size subset of  $P$  such that each point  $p \in C_p^*$  has distance from  $C_p^*$  less than a user-specified threshold  $\eta$ :

$$C_p^* = \arg \max_{C \subseteq \mathcal{P}} |\{p \in C : d(p, C) \leq \eta\}|. \quad (2)$$

Parameter  $\eta$  must be chosen so that two correct observations are at distance greater than  $\eta$  only with negligible probability. The justification for the above definition is that we want to include in  $C_p^*$  all observations  $p$  that best fit the underlying distribution. These observations should be such that, when considered together, the removal of one of them causes the loss of only a little information. Thus, we form  $C_p^*$  by discarding all those observations that are unlikely to be correct.

Finally, the *fault-tolerant estimate*  $\hat{\Theta}_{FT}$  is defined as the centroid of the fault tolerant cluster  $C_p^*$ :

$$\hat{\Theta}_{FT} = \text{centroid}(C_p^*). \quad (3)$$

Based on the above definitions, Fig. 4 provides a pseudocode of the proposed fault-tolerant cluster algorithm.

Figure 5 depicts an example in which the current cluster  $C$  comprises four points  $p_1, \dots, p_4$ , which are observations of a

<sup>3</sup>In information theory, the information of an event  $e$  is defined in terms of its probability of occurrence  $Pr\{e\}$ :  $I(e) = \log_2(1/Pr\{e\})$ .

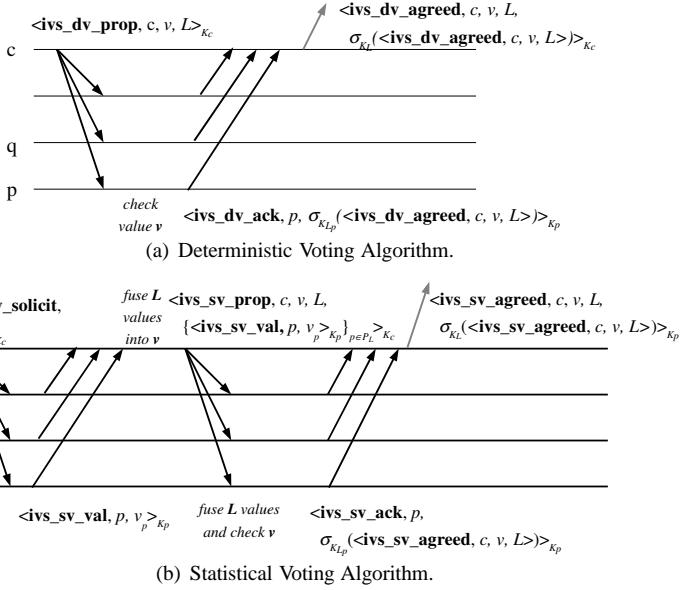


Figure 3. Inner-circle Voting Sequence

**Require:**  $L$  data points  $\{p_1, \dots, p_L\}$  and a threshold  $\eta$   
**Ensure:**  $\hat{\Theta}_{FT}$  is the fault-tolerant centroid  
1: // First compute the fault-tolerant cluster  $C$   
2:  $C := \{p_1, \dots, p_L\}$   
3:  $change := |C| > 2$   
4: **while**  $change$  **do**  
5:  $change := false$   
6: **for all**  $p_i \in C$  **do**  
7:  $\hat{\Theta}_i := \text{centroid}(C \setminus p_i)$   
8:  $d_i := \|p_i - \hat{\Theta}_i\|$   
9: **end for**  
10: **if**  $\exists p_i \in C : (d_i > \eta) \wedge (\forall p_j \in C : d_j \leq d_i)$  **then**  
11:  $C := C \setminus p_i$   
12:  $change := |C| > 2$   
13: **end if**  
14: **end while**  
15: // Then compute  $\hat{\Theta}_{FT}$  as the centroid of the fault-tolerant cluster  $C$   
16:  $\hat{\Theta}_{FT} := \text{centroid}(C)$

Figure 4. Fault-Tolerant Cluster algorithm.

common (unknown) value  $\Theta$  independently produced by four sensor nodes  $n_1, \dots, n_4$ . The current estimate  $\hat{\Theta}$  is computed considering all available data and has poor accuracy due to  $p_4$ —in the figure, we suppose that point  $p_4$  is due to an error in node  $n_4$ 's sensor, e.g.,  $n_4$ 's sensor was physically damaged by humidity and reports readings stuck at a high value. To decide whether to exclude point  $p_4$ , the fault-tolerant cluster algorithm first computes  $\hat{\Theta}_4$  as the centroid of points  $\{p_1, p_2, p_3\}$  and then computes the distance  $d_4$  between  $\hat{\Theta}_4$  and  $p_4$ . If distance  $d_4$  is greater than a user-supplied threshold  $\eta$ , then  $p_4$  is excluded from cluster  $C$ . In this case, the new estimate of  $\Theta$  is going to be  $\hat{\Theta}_4$ , which is much more accurate than  $\hat{\Theta}$ .

The fault-tolerant cluster algorithm cannot guarantee removal of (and only of) faulty/malicious data if these data are very similar to the correct data; however, the negative effect of such a case should be negligible. It can be shown that [16]: (1) If the number  $F$  of faulty/malicious points is less than half of the total points  $N$ , then condition  $\delta_F > \frac{\delta_C}{1-2F/N}$  guarantees that only faulty/malicious points are removed, where  $\delta_C$  and  $\delta_F$  represent the maximum distance from  $\hat{\Theta}_C$ —the estimate computed with only the correct points—in the correct and faulty/malicious points, respectively. (2) The worst-case sce-

**Deterministic Voting.** Node  $c$  proposes value  $v$  by sending a *propose message* to its inner-circle nodes ( $p$  and  $q$ ). Receiving node  $p$  checks the validity of  $v$  by evaluating  $f(v)$  and, in this case, sends an *ack message* to  $c$ , including a partial signature  $\sigma_{K_{lp}}$  obtained by using  $p$ 's share of secret  $K_L$ . On receiving  $L$  ack messages, node  $c$  generates its partial signature  $\sigma_{K_{lc}}$ , creates a total signature  $\sigma_{K_L}$ , and assembles an *agreed message* to be sent to the proper destination(s).

**Statistical Voting.** Node  $c$  proposes value  $v_c$  by sending a *solicit message* to its inner-circle nodes. Receiving node  $p$  replies with a *value message* that includes its proposed value  $v_p$ . On receiving  $L$  value messages,  $c$  computes the set  $P_L$  of  $L$  nodes that have sent their values, computes a fused value  $v = f(v_c, v_p, \dots)$ , and sends a *propose message* that includes value  $v$  and the value messages sent by each node  $p \in P_L$ . On receiving the propose message,  $p$  verifies that the included signatures are valid and that  $v$  was computed by applying  $f$  on the values  $\{v_c, v_p, \dots\}$ . In this case,  $p$  replies with an *ack message* that includes  $p$ 's partial signature  $\sigma_{K_{lp}}$ . On receiving  $L$  ack messages, node  $c$  assembles an *agreed message* to be sent to the proper destination(s).

In both algorithms, the actual forwarding in the wireless network occurs via the node's routing and forwarding services (see § 4). Eventually, a destination node receives the agreed message, verifies the included signatures, and delivers the included value to the local application.

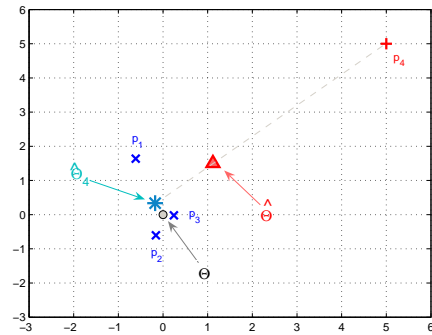


Figure 5. Fault-Tolerant Cluster example.

nario corresponds to all faulty/malicious observations clustering in a point  $p^*$  at a distance  $\delta_F^* = \frac{\delta_C}{1-2F/N}$  from  $\hat{\Theta}_C$ . Thus, the maximum estimation error  $E = \|\hat{\Theta}_C - \hat{\Theta}_{FT}\|$  added by faulty/malicious observations is  $E^* = \frac{F}{N} \delta_F^*$ . For instance, the case in which one third of the points are erroneous ( $F = N/3$ ) corresponds to  $\delta_F^* = 3\delta_C$  and  $E^* = \delta_C$ , which indicates that estimate  $\hat{\Theta}_{FT}$  is in the range of the correct observations.

## 5. Application Examples

The next sections demonstrate the inner-circle approach in two significant wireless application scenarios: (1) the neutralization of black hole attacks in AODV networks and (2) the neutralization of sensor errors in a wireless sensor network.

### 5.1. Reliable and Secure AODV Networks: Black Hole Attack Case Study

This section presents an application of the inner-circle approach to neutralize black hole attacks in AODV [2] wireless networks. In a black hole attack, a malicious node  $M$  advertises itself as having the shortest (or the most recent) path

to a node  $D$ , whose packets it wants to intercept. In AODV this can be achieved by replying to a received RREQ message with a malicious RREP message that has a large destination sequence number.<sup>4</sup> Once node  $M$  has been able to insert itself into an active routing path, it can drop packets to  $D$ . Black hole attacks are very difficult to detect and protect against because the mere use of user authentication and signed routing information cannot prevent compromised nodes from generating correctly signed yet malicious routing packets.

Some work has attempted to cope with black hole attacks in AODV networks. The proposed techniques require changes to AODV, have limited coverage and unbounded detection latency because they are based on network-wide mechanisms, and cannot cope with attack variations (gray hole attacks) in which a malicious node behaves most of the time as a good node and only sporadically as a black hole node [4, 5, 23].

In our approach, each wireless node embeds the inner-circle framework, which is configured to intercept incoming/outgoing RREP messages and to run the deterministic voting service (see § 4.2) to check the validity of those messages. The execution of the deterministic voting protocol is adapted to our case study by the instantiation (e.g., in a Linux shared library) of *Inner-circle Callbacks* (see § 4) that implement AODV-specific actions that prevent black hole attacks. This service is discussed in Fig. 6(a)–(d) through an execution example that assumes a dependability level  $L = 1$ . In Fig. 6(e), a malicious node  $M$  sends an invalid RREP message that never gets approved by  $M$ 's inner-circle nodes, and thus, never propagates in the network.

It is possible to show [16] that if the dependability level  $L$  is chosen so as to guarantee at least one non-Byzantine inner-circle node other than the center node (i.e.,  $T = 1$ , see § 4.2), then the proposed mechanism guarantees that only valid routes are established, i.e., it is impossible for a malicious node  $M$  to diffuse a malicious RREP message for a destination  $D$  if  $M$  is not on a path to destination  $D$ .

We have used the ns-2 network simulator [3] to study the effectiveness of the proposed inner-circle approach in neutralizing black hole attacks. The simulation parameters and the results are reported in Fig. 7. Figure 7(a) shows the overall network throughput (measured as the total number of packets received in the network divided by the total number of packets sent in the network). A significant result is that a single malicious node is capable of inflicting about 1000% throughput degradation (from 98% throughput with no attack to 9% throughput with attacks) in a network of 50 nodes. This degradation goes up to 3000% (3.5% throughput) for 10 malicious nodes. On the other hand, the inner-circle approach pays the price of a 10% throughput degradation in the absence of attacks (due to underlying STS and IVS communication) but can significantly reduce the effect of the malicious nodes to only 22% throughput degradation. Figure 7(b) shows a node's average energy consumption. The results indicate that the

Simulation Parameters	
Topology:	1000 × 1000 m <sup>2</sup>
Number of nodes:	50
Node mobility:	Random waypoint model
	Speed 10 m/s, Pause time 0 s
Traffic:	Constant Bit Rate UDP traffic
	Send rate: 4 packets/s
	Packet size: 512 bytes
Number of connections:	10
Physical layer:	802.11
Transmission range:	250 m
Energy consumption:	
	Tx 660 mJ, Rx 395 mJ, Idle 35 mJ
Simulation time:	300 s
Inner-circle framework:	
	STS timer: $\Delta_{STS} = 2$ s
	Dependability level: $L = 1, 2$
	Secret key length: 1024 bits
Number of experiments:	50

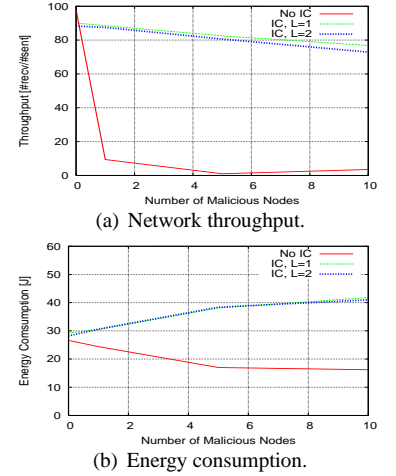


Figure 7. Simulation study of black hole attack.

overhead brought by the inner-circle approach ranges from 7% in the absence of attacks, to less than 50% in the presence of attacks. Note that when the inner-circle approach is not used, black hole attacks result in reduced energy consumption because of the reduced number of messages delivered in the network.

## 5.2. Reliable and Secure Sensor Networks: Faulty Sensors Case Study

This section demonstrates an application of the inner-circle consistency approach to improve sensor data accuracy in spite of sensor errors. The scenario considered is a wireless sensor network deployed in a remote region  $R$  to detect and localize events of interest. It is assumed that a target event at a location  $u$  emits an energy signal  $S_i(u)$  that can be measured by a sensor node  $i$  at location  $s_i$ . In addition, the strength of the emitted signal is assumed to decay polynomially with the distance, as modeled below [19]:

$$S_i(u) = \begin{cases} K \cdot T & \text{if } d < d_0; \\ \frac{K \cdot T}{(d/d_0)^{\epsilon}} & \text{otherwise,} \end{cases} \quad (4)$$

where  $K$  is the power emitted at the target's location  $u$ ,  $T$  is the sensor's sampling duration,  $d = \|u - s_i\|$  is the distance between the target and the sensor, and  $d_0$  is a constant determined by the physical sizes of the target and of the sensor. Since a sensor's energy measurements are usually corrupted by random environmental/measurement noise, the total energy measured by sensor  $i$  is assumed to be  $E_i = S_i(u) + N_i^2$ , where  $N_i \sim \mathcal{N}(0, \sigma_N)$ .

The task of each sensor node is to detect the presence of a nearby target and to estimate the target's position. To detect the presence of targets, sensor nodes follow the Neumann-Person strategy: sensor  $i$  detects a target if the sensed energy  $E_i$  is greater than a predetermined threshold  $\lambda$ , a parameter chosen to maximize the detection probability  $P_D$  while keeping the false alarm probability  $P_F$  below a desired threshold  $\alpha$ .

<sup>4</sup>Both RREQ and RREP messages include a *destination sequence number* that is used to distinguish fresher routes from older ones.

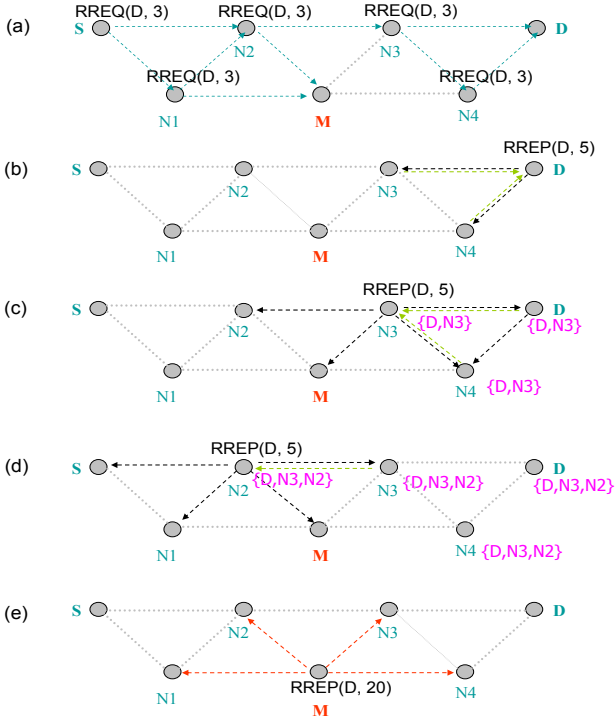


Figure 6. Neutralizing a black hole attack.

To localize a detected target, sensor nodes use their own position  $s_i$  as estimations. Once a sensor  $i$  detects and localizes a target, its task is to send a target notification  $\langle t_i, E_i, u_i \rangle$  to the base station, where  $t_i$  is the detection time,  $E_i$  is the energy level with which the target was sensed, and  $u_i$  is the target's estimated position. The directed diffusion protocol stack is used to support this communication [14].

Sensor devices interact directly with the environment and, hence, are subjected to a variety of physical, chemical, and biological forces; this makes them degrade fairly quickly. Field studies [6] indicate that errors originating in degraded sensor devices are a major cause of unreliability in a wireless sensor network. Interestingly, these sensor failures are likely to manifest days before the sensor electronics may fail. Based on these results, we assume the following *sensor fault model*:

- *Stuck at Zero*, a faulty sensor  $i$  constantly reports a fixed reading (a zero reading in this study):  $E_i = 0$ ;
- *Calibration Error*, a faulty sensor  $i$ 's readings are affected by a multiplicative error:  $E_i = \epsilon_{clbr} \cdot (S_i + N_i^2)$ ;
- *Signal Interference*, a faulty sensor  $i$  reports readings affected by strong environmental disturbances:  $E_i = S_i + \epsilon_{intf} \cdot N_i^2$ , with  $\epsilon_{intf} \gg 1$ ;
- *Positioning Error*, a faulty sensor  $i$  has an incorrect estimate of its own position  $s_i$ :  $s_i \sim \text{Uniform}(R)$ .

The remainder of this section contrasts two solutions to the detection/localization problem formulated above: (1) a *centralized solution*, where the base station collects raw target notifications  $\langle t_i, E_i, u_i \rangle$  as they are generated by the sensor nodes;

In AODV, a source  $S$  requests a route to a destination  $D$  by flooding a RREQ message in the network. When the RREQ reaches  $D$ , node  $D$  constructs a RREP, which is unicast back to  $S$  using the reverse route through which RREQ was received. Forwarding nodes update their routing tables so as to create a route from  $S$  to  $D$ .

In the inner-circle approach, the Inner-circle Callbacks maintain a mapping  $fw$  in each node. The mapping associates a pair  $(D, dseqno)$ —where  $D$  is a destination node and  $dseqno$  is a destination sequence number—with the set of nodes (depicted in brackets in the figure on the left) that can forward messages addressed to node  $D$  when the active route to  $D$  has destination sequence number  $dseqno$ . The operation is as follows:

- AODV service at a node  $c$  (initially node  $D$ ) sends a RREP message  $\langle \text{RREP}, route\_dst, dseqno, next\_hop \rangle$ , where  $route\_dst$  is destination  $D$ ,  $dseqno$  is the associated destination sequence number, and  $next\_hop$  is  $c$ 's designated next node in the process of unicasting the RREP message back to node  $S$ .
- Node  $c$ 's Inner-circle Interceptor intercepts the RREP message and passes it to  $c$ 's Inner-circle Voting Service (see Fig. 1). At this point, a deterministic voting algorithm is executed (see Fig. 3(a)) in which node  $c$  sends to its inner-circle nodes a propose message that includes the intercepted RREP message.
- On receiving the propose message, an inner-circle node  $p$  verifies that  $c$ 's proposed RREP is valid (method `check` of Inner-circle Callbacks). The check succeeds only if  $c$  is the destination of the sought route ( $c = route\_dst$ ) or  $p$  considers  $c$  as a valid forwarding node to the route destination ( $c \in fw(route\_dst, dseqno)$ ).
- If  $L$  inner-circle nodes acknowledge  $c$ 's proposed RREP, then  $c$  assembles an agreed message and sends it to all its inner-circle nodes (including  $c$ 's designated next hop).
- On receiving the agreed message, an inner-circle node  $p$  updates the local mapping  $fw$  to include both nodes  $c$  and  $next\_hop$  (method `onAgreed`); moreover, if  $p$  is  $c$ 's designated  $next\_hop$ , then  $p$  passes the RREP message encapsulated in the agreed message to its local AODV service.
- The operation continues with node  $p$ 's AODV service sending an RREP message heading back to  $S$ .

and (2) an *inner-circle solution*, where each wireless sensor node embeds the inner-circle framework, which is configured to intercept incoming/outgoing directed diffusion messages that carry target notifications  $\langle t_i, E_i, u_i \rangle$ , and to run the statistical voting service. The execution of the statistical voting protocol is adapted to our case study by the instantiation (e.g., in a TinyOS component) of Inner-circle Callbacks (see § 4) that implement sensor-specific actions to prevent both faulty and redundant data propagation. Due to space limitations, formal specifications of these actions are relegated to [16].

To carry out reliable and secure in-network processing, the inner-circle solution uses statistical voting to improve the fidelity of each field of a target notification  $\langle t_i, E_i, u_i \rangle$ , where the fault-tolerant cluster algorithm<sup>5</sup> of § 4.3 is used as the fault-tolerant fusion function. In particular, a target's position is estimated locally by (1) computing the distance  $d_i$  of each inner-circle sensor  $i$  from the target (by using Eqn. 4), (2) using a trilateration algorithm on each triple of pairs  $(u_i, d_i)$  to obtain target location estimates  $p_i$ , and (3) filtering the obtained  $\binom{L}{3}$  estimates  $p_i$  with the fault-tolerant cluster algorithm.

We have used the ns-2 network simulator [3] to study the effectiveness of the proposed inner-circle approach in coping with sensor errors. The simulation parameters and the results are reported in Fig. 8. Figure 8(a) shows the probability of missing a valid target, which is zero for all configurations considered. Figure 8(b) shows the probability of a spurious target detection. In the centralized case (marked as “No IC” on all graphs), this probability can be as high as 19% (under the signal interference fault model), while the inner-circle

<sup>5</sup>Parameter  $\eta$  is set based on the standard deviation of noise  $N_i^2$ .

Simulation Parameters	
Topology:	$200 \times 200 m^2$
Number of nodes:	100
No node movement	
Sensor Model:	
Target sensing rate:	5 s
Neumann-Person threshold:	$\lambda = 6.635$
Single node maximum $P_F$ :	$\alpha = 0.01$
$K \cdot T =$	20000
$k = 2$	$\sigma_N = 1$
Single target of 25s duration every 100s	
Number of faulty nodes:	10
Sensor fault model:	$\epsilon_{clbr} = 2, \epsilon_{inf} = 10$
Physical layer:	802.11
Transmission range:	40 m
Energy consumption:	
Tx 660 mJ, Rx 395 mJ, Idle 35 mJ	
Simulation time:	200 s
Inner-circle framework:	
STS timer: $\Delta_{STS} = 100$ s	
Dependability level: $L = 2..7$	
Secret key length:	512 bits
Fault-tolerant cluster threshold:	$\eta = 5$
Number of experiments:	50

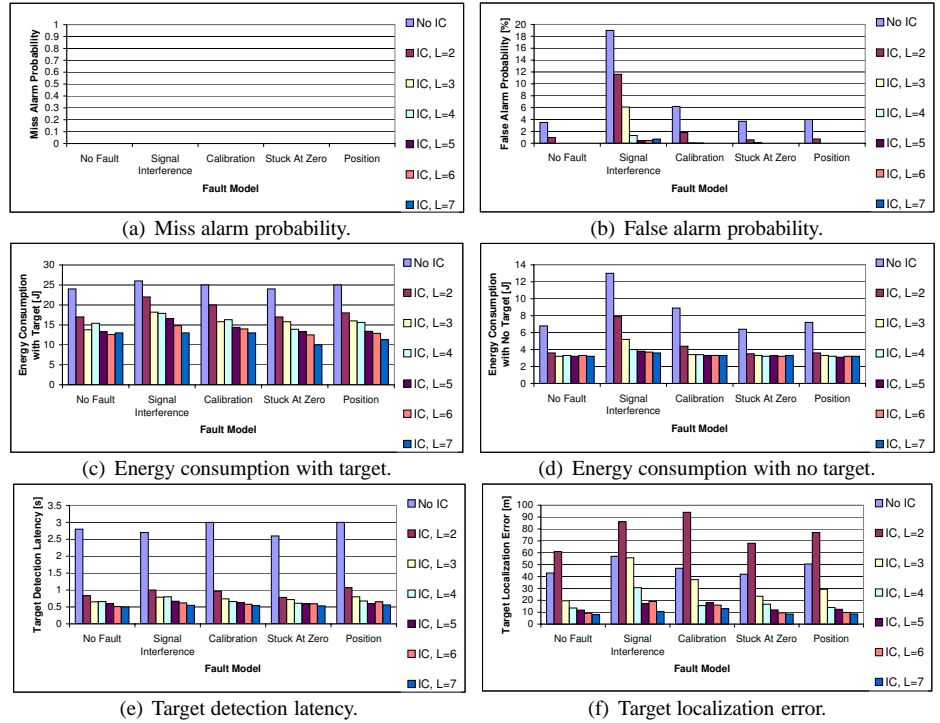


Figure 8. Simulation study of a faulty sensor network.

solution can reduce it to zero by exploiting target information shared by neighboring nodes. Figure 8(c) and Fig. 8(d) show a node’s average energy consumption both when there is a target and when there is no target. The figures indicate an over 50% reduction in energy consumption (due to the ability of suppressing both duplicate and spurious detections), which can result in doubling overall network lifetime. Figure 8(e) and Fig. 8(f) show the target detection latency (i.e., the time elapsed between when a target pops up and when the base station receives the first notification) and the target localization error (i.e., the distance between the real target location and the location estimated by the sensor network), respectively. The inner-circle solution provides a six-fold reduction in detection latency, and between a four-fold and five-fold reduction in localization error (for inner-circle sizes over four nodes).

In conclusion, the inner-circle approach provides notably improved performance in the considered configuration and fault models. While this result is significant, we cannot expect the same improvement if node density is sparse or, alternatively, if the signal emitted by the target is weak and is detected only by very few neighboring nodes (e.g., 1–2). To study this point further, we have re-run the experiments while using a weaker target signal ( $K \cdot T = 10000$ ). Apart from the miss alarm probability, all considered metrics show an improvement similar to that of Fig. 8. The miss alarm probability, however, increases up to 2-5% for inner-circle sizes greater than five nodes, where the worst cases correspond to signal interference and stuck-at-zero fault modes.

## 6. Related Work

Providing availability despite crashes and node movement has been investigated by a number of researchers who have attempted to migrate fault tolerance techniques for distributed systems (e.g., replication, group communication, checkpointing) to the wireless domain [24–26]. Because of their complexity (heavy interaction between non-neighbor nodes), these techniques have rarely been extended to more complex failure models (e.g., data corruption, Byzantine faults) [27].

Providing security guarantees (e.g., integrity, authentication, authorization) is the goal of a significant body of literature in wireless ad hoc networks (including wireless sensor networks). However, only a few studies have started to investigate the protection of wireless environments against internal attacks [28, 29].

In [29], intrusion tolerance is attempted by means of admission control and threshold cryptography. In order to participate in the network, a node must acquire a token (in practice, a certificate) from its neighbors. The token is collectively signed by  $K$  neighbors (through threshold cryptography) and must be renewed periodically. The scheme presupposes a local intrusion-detection module running on each node and, under the assumption that a malicious node  $n$  is eventually detected by a sufficient number of its neighbors, guarantees that malicious node  $n$  is eventually denied access to the network (its neighbors will revoke its token). The scheme fails if a malicious node changes its neighborhood before being detected and convicted by its previous neighbors. Consequently, the network may be left vulnerable to attacks for arbitrarily long periods—consider, for instance, a malicious node  $n$  that in-

jects malicious attacks intermittently while roaming the network. It could be argued that the proposed inner-circle approach shares notions of threshold cryptography and localized message signing with [29]. A significant difference, though, is to be found in the masking nature of inner-circle voting, which prevents erroneous data propagation and, hence, avoids the problems of [29].

The concept of intrusion and fault tolerance has been extensively studied in the context of distributed systems and has resulted in a number of prototypes [15, 30, 31]. The inner-circle consistency approach differs substantially from these techniques, as it is specifically designed for mobile wireless environments. Typical intrusion and fault tolerance techniques forcibly replicate data and computation on multiple nodes in order to mask errors/intrusions in a server application run on these nodes. In contrast, inner-circle consistency leverages the partially replicated data/computation that is naturally available in a locality of a sender node to neutralize errors/attacks originating from that node.

## 7. Conclusions

This paper proposes the notion of *inner-circle consistency* to protect wireless ad hoc networks from errors and attacks. Through local node interaction, errors/attacks are neutralized at the source, both preventing their propagation in the wireless network and improving the fidelity of the propagated information. Thus, an unreliable and insecure wireless network is transformed into a dependable network substrate on top of which applications benefit from improved network reliability and security. This goal is achieved by combining statistical (a proposed *fault-tolerant cluster* algorithm) and threshold cryptography techniques with application-aware checks to exploit the data/computation that is partially and naturally replicated in wireless applications. A formally specified inner-circle framework is prototyped with the ns-2 network simulator and used to demonstrate the idea of inner-circle consistency in two significant wireless scenarios: (1) the neutralization of black hole attacks in AODV networks and (2) the neutralization of sensor errors in a target detection/localization application executed over a wireless sensor network.

## Acknowledgments

This work is supported in part by NSF grant CCR 00-86096 ITR, MURI grant N00014-01-1-0576, the Gigascale Systems Research Center (GSRC/MARCO), and the Motorola Corporation as part of Motorola Center. We thank Fran Baker for insightful editing of our manuscript.

## References

- [1] C. Basile, M.-O. Killijian, and D. Powell. A survey of dependability issues in mobile wireless networks. Technical report, LAAS-CNRS, Toulouse, 2003.
- [2] C. E. Perkins, E. M. Belding-Royer, and I. Chakeres. Ad Hoc On Demand Distance Vector (AODV) Routing. Technical report, IETF Internet draft, 2003.
- [3] *The Network Simulator — ns-2*. <http://www.isi.edu/nsnam/ns/>.
- [4] W. Wang and B. Bhargava. On vulnerability and protection of ad hoc on-demand distance vector protocol. In *Proc. of Int'l Conf. on Telecommunication*, 2003.
- [5] S. Ramaswamy, H. Fu, and M. Sreekantaradhya. Prevention of cooperative black hole attack in wireless ad hoc networks. In *Proc. of Int'l Conf. on Wireless Networks*, 2003.
- [6] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Proc. of European Workshop on Wireless Sensor Networks*, 2004.
- [7] C. Savarese, J. Rabay, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *Proc. of USENIX Technical Conference*, 2002.
- [8] V. Shoup. Practical threshold signatures. *LNCS*, 1807:207–218, 2000.
- [9] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. *LNCS*, 963, 1995.
- [10] F. Cristian and C. Fetzer. The timed asynchronous distributed system model. *IEEE Trans. on Parallel and Distributed Systems*, 10(6), 1999.
- [11] J. Douceur. The sybil attack. In *Proc. of the IPTPS*, 2002.
- [12] L. Sankaranarayanan, G. Kramer, and N. Mandayam. Hierarchical sensor networks: capacity bounds and cooperative strategies using the multiple-access relay channel model. In *Proc. of SECON*, 2004.
- [13] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. on Prog. Languages and Systems*, 4(3), 1982.
- [14] C. Intanagonwivat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proc. of MobiCom*, 2000.
- [15] G. P. Saggese, C. Basile, L. Romano, Z. Kalbarczyk, and R. Iyer. Hardware support for high-performance, intrusion- and fault-tolerant systems. In *Proc. of SRDS*, 2004.
- [16] C. Basile, Z. Kalbarczyk, and R. Iyer. Neutralization of error and attacks in wireless ad hoc networks. Technical report, University of Illinois at Urbana-Champaign, 2005.
- [17] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. of TACAS*, 1996.
- [18] D. Dolev et al. Reaching approximate agreement in the presence of faults. *J. of the ACM*, 33(3):499–516, 1986.
- [19] T. Clouqueur, K. K. Saluja, and P. Ramanathan. Fault tolerance in collaborative sensor networks for target detection. *IEEE Trans. on Computers*, 53(3):320–333, 2004.
- [20] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *J. of the ACM*, 32(1):52–78, 1985.
- [21] M. H. Azadmanesh and R. M. Kieckhafer. New hybrid fault models for asynchronous approximate agreement. *IEEE Trans. on Computers*, 45(4):439–449, 1996.
- [22] L. Kleinrock and J. Silvester. Optimum transmission radii for packet radio networks or why six is a magic number. In *Proc. of IEEE National Telecommunications Conference*, 1978.
- [23] H. Deng, W. Li, and D. P. Agrawal. Routing security in wireless ad hoc network. *IEEE Communications Magazine*, 2002.
- [24] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proc. of INFOCOM*, pages 1568–1576, 2001.
- [25] Q. Huang, C. Julien, G. Roman, and A. Hazemi. Relying on safe distance to ensure consistent group membership in ad hoc networks. In *Proc. of ISCE*, 2001.

- [26] R. Prakash and M. Singhal. Low-cost checkpointing and failure recovery in mobile computing systems. *IEEE Trans. on Parallel Distrib. Syst.*, 7(10):1035–1048, 1996.
- [27] J. Luo, J.-P. Hubaux, and P. Eugster. PAN: Providing reliable storage in mobile ad hoc networks with probabilistic quorum systems. In *Proc. of MobiHoc*, pages 1–12, 2003.
- [28] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of Int'l Conf. on Mobile Computing and Networking*, pages 255–265, 2000.
- [29] X. M. H. Yang and S. Lu. Self-organized network layer security in mobile ad hoc networks. In *Proc. of MOBICOM*, 2002.
- [30] L. Zhou et al. Coca: A secure distributed online certification authority. *ACM Trans. on Computer Systems*, 20(4), 2002.
- [31] MAFTIA Project. <http://www.newcastle.research.ec.org/maftia/>, 2003.