Cryptographic Protocols to Fight Sinkhole Attacks on Tree-based Routing in Wireless Sensor Networks

Anthonis Papadimitriou
INRIA
anthony@di.uoa.gr

Fabrice Le Fessant
INRIA
fabrice.le_fessant@inria.fr

Aline Carneiro Viana
INRIA
aline.viana@inria.fr

Cigdem Sengul
DT Labs, TU-Berlin
cigdem.sengul@telekom.de

Abstract—This work introduces two new cryptographic protocols of different complexity and strength in limiting network degradation caused by sinkhole attacks on tree-based routing topologies in Wireless Sensor Networks (WSNs). The main goal of both protocols is to provide continuous operation by improving resilience against, rather than detection of, these attacks. The main benefit of providing resilience is that it allows operating (or graceful degradation) in the presence of attacks. Furthermore, while resilience mechanisms do not dismiss detection mechanisms, detection mechanisms often introduce more complexity and so, more weaknesses to the system, which might not justify their benefits. We provide a simulation study of the two protocols for three different routing protocols, that encompass typical routing strategies used in WSN. The results of our simulation study show that our cryptographic protocols are effective in improving resilience against sinkhole attacks, even in the presence of some collusion.

I. Introduction

Wireless Sensor Networks (WSN) are penetrating more and more in our daily life. As a consequence, security has become an important matter for these networks. Indeed, many WSN protocols are not secured even against simple attacks. Let's take a simple scenario for a possible attack: a supermarket is protected during closing hours by a WSN, where each sensor is able to sense any movement in its neighborhood, and report an alarm to the sink at the center of the supermarket. A thief can put a sensor in the neighborhood of the supermarket, that tells sensors in the supermarket that it is closer to the sink (i.e., *sinkhole* attack). Hence, all their messages destined to the sink, especially alarms, will be routed through that sensor, which will discard alarm messages (i.e., *selective forwarding* attack) and let the thief enter the supermarket.

Recent work [1], [2] has mostly focused on how such an attack can be detected with high probability. However, attack detection often increases the complexity of the system and thus, increases the possibility of other attacks. Furthermore, fault-detection mechanisms or network debuggers can also be effected by sinkhole attacks [3]. In [3], a network debugger is proposed, which requires periodic metric collection at the sink. The sink decides which nodes have delivered sufficient data and if not, infers the causes of the failures. However, during their evaluations, the authors ran into a problem: some nodes had a route to a node that was not the sink. The authors validated that the network was exposed temporarily to an extraneous node with the same ID as a node already in the network. That node advertised itself as a sink, confusing

the network. From that point on, due to a bug in the routing code, the network never recovered. This example shows that ensuring built-in *resilience* to sink-hole attacks would be more effective. Consequently, we propose two *RESIlient and Simple Topology-based reconfiguration protocols*: RESIST-1 and RESIST-0. RESIST-1 prevents a malicious node from modifying its advertised distance to the sink by more than one hop, while RESIST-0 does not allow such lying at the cost of additional complexity.

We studied the performance of RESIST-1 and RESIST-0 through simulations, for three tree-based routing protocols that encompass different routing strategies. Our simulation results show that our reconfiguration protocols are effective in improving resilience against sinkhole attacks in WSNs. Finally, we conclude with a discussion on implementation issues, such as the use of cryptography in sensor networks, and on the impact of collusion on our protocols.

The remainder of this paper is structured as follows. In Section II we present the system model. In Section III, we lay out our proposal for two simple and resilient topology-based routing protocols. Performance results are presented in Section IV and discussions in Section V. The Section VI overviews the current literature. Finally, Section VII concludes with future work.

II. System Model

This section presents the network and the threat models, as well as, our risk metric, the Risk Factor.

A. Network model

We consider a WSN consisting of a set V of N static sensors, including *only* one sink. Nodes have unique IDs and are randomly scattered on a geographical area. All sensors are similar in terms of computational, memory, and communication capabilities. Nodes do not need access to location information. Each node X is able to communicate wirelessly with a subset of nodes (its *neighbors* N_X) that are in its transmission range, r_t. We consider a large set of routing protocols relying on tree-based topology construction [4]–[10]. The data is thus, routed from sensor nodes to the sink through a tree rooted at the sink. The routing tree is an aggregation of the shortest paths from each sensor to the sink based on a cost metric, which can represent different application requirements

(e.g., cost hop count, loss, delay). In this work, the routing tree is built by using the hop distance to the sink.

B. Threat model

We focus on sinkhole attacks launched by compromised nodes inside the network. In our threat model, sensors cannot lie about their identities due to the presence of cryptographic measures [11]. In general, we assume malicious sensors are not colluding (i.e., collaborating to increase the impact of the attack). The impact of collusion is discussed in Section V.

We assume that public-key cryptographic primitives are available on all sensors (refer to [12] for a survey on key distribution in WSNs). Recently it has also been shown that public key infrastructure is viable for WSNs [13]. Hence, in our model, all sensors only know and trust the public key K_{pub}^{sink} of the sink. Additionally, each sensor X has a pair of public-private keys (K_{pub}^X, K_{pri}^X) that it can use to prove its identity. These key pairs can be generated and uploaded offline to the sensors before the deployment. Using these key pairs, nodes perform authentication and sign data messages.

C. Risk model

In this paper, we use a new and more comprehensive metric, called *Risk Factor* [14] to measure the strength of an attack. Intuitively, the Risk Factor is the probability that a message reaches compromised nodes (that might drop it) on its route to the sink, and inherently captures parameters such as: the number of compromised sensors, their position, the density, and the size of the network. It is computed iteratively starting from the sink using the following formula:

$$NodeRisk_X = \begin{cases} 0 & \text{if } X \text{ is the sink} \\ 1 & \text{if } X \text{ is malicious} \\ \frac{\sum_{Y \in N_X | d_Y < d_X} NodeRisk_Y}{\|\{Y \in N_X | d_Y < d_X\}\|} & \text{if } X \text{ at distance } d_X, \end{cases} \tag{1}$$

$$RiskFactor = \frac{\sum_{X \in V} NodeRisk_X}{\|V\|} \tag{2}$$

We evaluated Risk Factor with different number of compromised nodes using 3 different positioning strategies: The compromised nodes are placed (1) around the sink (ring placement) at a distance equal to the transmission range of the sink (denoted as R), (2) using ring placement at a distance $1.5R$ and (3) randomly (random placement). Fig. 1 illustrates the representativeness of the Risk factor compared to evaluating the risk using "the number of compromised nodes" as a metric. For instance, the Risk Factor is able to capture that surrounding the sink at a closer distance performs better than random placement. However, as the ring is placed further away than the sink, random placement brings higher risks as the number of nodes increases. On the other hand, the damage of a ring placement is limited as the number of nodes that can be packed to a ring around the sink is also limited. Note that using "the number of compromised nodes" the risk would increase regardless of how these nodes are positioned. However, most of the time, the risk increases more slowly or remains constant after a few compromised nodes.
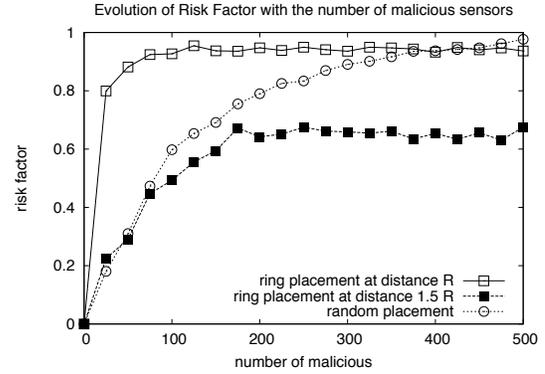

Fig. 1. The Risk Factor grows very fast with the first compromised nodes, and more slowly afterwards. A few well-positioned nodes (a ring at distance R) can impact the network as much as hundreds of nodes randomly positioned.

III. SECURITY PROTOCOLS

To achieve higher resilience in tree-based routing protocols [4]–[10], we propose two protocols. These protocols are used at the tree reconfiguration phase triggered by the sink. Note that we do not have special constraints on the period between reconfigurations: it can be chosen either similar to the current tree-based routing protocols, based on cost, or topology vulnerability. We define a class of RESIST-h reconfiguration protocols that allow malicious nodes to modify their advertised distance to the sink, but no more than h hops. Based on this definition, we introduce two protocols, RESIST-1 and RESIST-0, which are presented in the remainder of this section. We also describe here cryptographic operations and message contents of the proposed protocols, but refer the reader to Section V for an efficient way of implementing them.

A. Simple reconfiguration protocol (RESIST-1)

The reconfiguration starts by the sink sending a Hello($epoch, tokens$) message to all its neighbors, where $epoch$ is a strictly increasing timestamp, chosen by the sink and $tokens$ is a list of tokens $[T_0, T_1, T_2, ..., T_R]$. Essentially, each token is a *(k, epoch)* pair signed by the sink, where k is the token number:

$$T_k = (k, epoch)_{signed(K_{pri}^{sink})} \tag{3}$$

When a sensor receives a Hello message, and after verifying that the tokens are correctly signed by the sink (i.e. verified by using the public key K_{pub}^{sink}), it does the following:

- If the *epoch* is new, it remembers the identity of the node sending it (i.e., his *parent*), and propagates the Hello message after removing the smallest token from the list of tokens. In other words, it receives Hello($epoch, [\mathbf{T_k}, T_{k+1}, ..., T_R]$) but sends Hello($epoch, [\mathbf{T_{k+1}}, ..., T_R]$).
- If the *epoch* is already known, but the Hello message advertises a shorter hop distance to the sink (i.e., contains a smaller token), the node might follow different approaches. A *selfish approach* would only update the node itself, while a *gossip approach* would also propagate

a new `Hello` message to the neighbors. In the rest of the paper, we follow the gossip approach.

Each sensor remembers as its *parent*, from which it received the smallest token signed by the sink for the most recent epoch. Note that the token number of the smallest token is also the hop distance to the sink. Alternatively, sensors can also remember *all the nodes* that advertise the shortest distance for a given epoch. In Section IV, we also evaluate this approach.

Sinkhole attack resilience: A compromised node can directly forward a `Hello` message without dropping the first token. Assume that the node is the first compromised node on the branch that the `Hello` message travels. Then, if the compromised node is at distance k from the sink, its neighbors would believe they are at distance k too, and so they would believe that the compromised node is at distance $k - 1$. Nevertheless, the compromised node cannot pretend to be at a distance smaller than $k - 1$, because it would be unable to provide smaller tokens than T_k. Note that as the `Hello` message travels down the tree, it might encounter other malicious nodes that do not drop the token before forwarding the message. In this case, each sensor would believe to be at a shorter distance to the sink depending on how many malicious nodes exist before it (e.g., if this number is 2, then it believes it is 2 hops closer to the sink than the reality). However, this might not degrade the performance significantly because the main impact is caused by the malicious node closest to the sink.

B. Complex reconfiguration protocol (RESIST-0)

To provide higher resilience, we next propose a more complex reconfiguration protocol (RESIST-0). This protocol is inspired by a protocol used to measure availability in peer-to-peer networks [15], where newly generated pairs of cryptographic keys are diffused in the network at every round. The sink sends a $\texttt{Hello}(epoch, [T_0, T_1, ..., T_R])$ message, where the generated tokens are:

$$T_k = ((k, epoch, K_{pub}^k)_{signed(K_{pri}^{sink})}, (K_{pri}^k)_{signed(K_{pri}^{sink})}), \quad (4)$$

where (K_{pub}^k, K_{pri}^k) is a newly generated pair of cryptographic keys for token k at a reconfiguration $epoch$. The reconfiguration protocol is the same as RESIST-1, except that, at the reception of a new epoch and before choosing a sensor Y as its parent, a sensor X challenges the sensor Y first by sending a $\texttt{Challenge}(k, epoch)$ message. Basically, this message asks Y to prove its distance k from the sink (i.e. that it has a copy of the token T_k). Sensor Y replies with a message `ChallengeReply`, which contains:

$$((k, epoch, K_{pub}^k)_{signed(K_{pri}^{sink})}, (ID^Y, ID^X)_{signed(K_{pri}^k)})$$

$(k, epoch, K_{pub}^k)_{signed(K_{pri}^{sink})}$ is the first half of the token T_k that Y received. At the reception of the `ChallengeReply` message and using the public key K_{pub}^{sink}, node X can first verify if the token k was correctly signed by the sink. In addition, node X recovers the public key of the token k, K_{pub}^k. Then, it can decrypt the second part of the `ChallengeReply` message, which was encrypted by the private key of token k, K_{pri}^k. Node X can thus verify if its identity, ID^X, was correctly signed by node Y. If so, it believes in ID^Y and in the Y's advertised distance k from the sink.

Sinkhole attack resilience: Since a sensor can sign the second part of the `ChallengeReply` message, if and only if it knows the private key for the token k, it is impossible for a compromised sensor (without collusion) to correctly reply to a `Challenge`. Furthermore, compromised nodes cannot even carry out the attack that we described for RESIST-1. Essentially, not dropping the smallest token would fail, because they would not be able to respond to the `Challenge` for the shorter hop count. Hence, RESIST-0 provides strong resilience against sinkhole attacks. We discuss the impact of collusion on our protocols in Section V.

IV. PERFORMANCE EVALUATION

The goal of our evaluation is to measure the amount of resilience obtained by RESIST protocols described in Section III. To this end, we present performance results under malicious attacks using three baseline routing protocols, described hereafter. The experiments were run in a discrete event-based simulator implemented in Java. As we are only interested in a RESIST's algorithmic evaluation, our simulator uses a simplified MAC layer, where neither message losses, nor collisions are considered. The correct operation of the presented schemes are independent of the order of message arrival. Furthermore, `Hello` message losses are typically handled by tree generation algorithms. On the other hand, lost `Challenge` or `ChallengeReply` might delay tree generation but do not jeopardize resilience.

A. Simulation Setup

This section describes the three baseline protocols and our simulation setup. We consider a data collection application, where each sensor only sends data (e.g., measurements) to the sink. The routing topology to reach the sink is regularly reconfigured [10]. Malicious nodes do not generate data and they drop every received message with probability $p = 1$. In addition, these nodes try to attract higher volumes of traffic by advertising shorter paths. We studied the performance of these protocols in networks when resilient reconfiguration schemes are used (RESIST-1 and RESIST-0) and not used (*vulnerable*).

In our simulator, we implemented three baseline routing protocols: *FTree*, *RRobin* and *RWalk*. In *FTree*, the routing tree is built once at each topology reconfiguration phase. *RRobin* differs from *FTree* as each sensor computes a set of alternative parents during the topology reconfiguration. This set includes the neighbor that sent the first `Hello` message and any neighbor that sent a `Hello` message with a hop count smaller or equal to the first neighbor. Each time a sensor has to send a message, it selects one parent from this set in a round robin way. In *RWalk* protocol, each sensor makes a random decision about forwarding a message either over the routing tree (computed as in *FTree*) or forwarding it to a randomly selected neighbor. If the message is not sent over the tree, it follows a n-hop random walk and after n hops, it is again forwarded over the tree. The goal of both *RRobin* and *RWalk* protocols is to allow escaping regions that may be
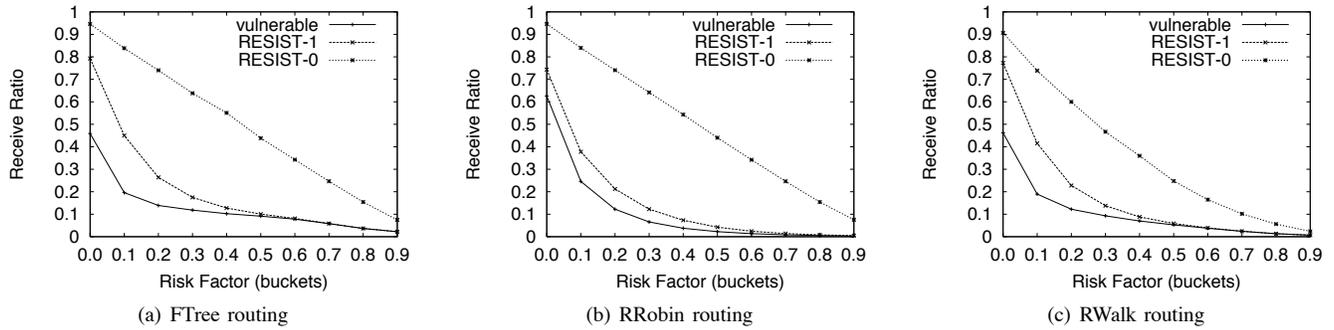
Fig. 2. Performance gain for different routing protocols: (a) Fixed Tree, (b) Round-Robin with, (c) Random-Walk ($n = 1$)
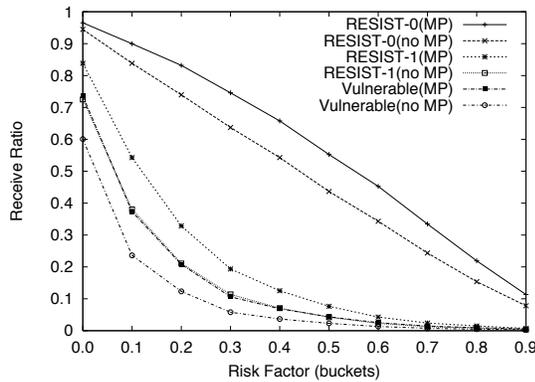

Fig. 3. Using two paths (MP) instead of one increases the overall performance of the different strategies.

severely affected by malicious nodes. In all our experiments with *RWalk*, we use $n = 1$.

We generated many random topologies, progressively filling them with malicious sensors. The space of topologies was divided in 10 *buckets*, where buckets 0-9 contain the topologies whose risk factor is in [0,0.1)-[0.9, 1.0), respectively. At each step, the risk factor was evaluated and the topology added to the corresponding bucket, until every bucket had at least 100 topologies. Using these topologies, the performance gain was computed as the ratio of messages that actually reach the sink compared to the number of messages that should reach the sink if no sensor were compromised.

B. Resilient Reconfiguration Protocols

Our results show that RESIST-0 achieves significant performance gain for all routing protocols (see Fig. 2). For both vulnerable and RESIST-1 cases, the decrease of the receive ratio is roughly exponential, whereas for RESIST-0, it has a better, linear decrease, as it does not allow nodes to lie about their distance to the sink. Fig. 2 confirms that when malicious sensors are able to lie, they can attract more network traffic and thus, incur a much higher impact in the WSN. The linear decrease in performance of RESIST-0 seems to be the upper bound of the performance we can obtain by only addressing the sinkhole attacks. To get better results, one must also fight selective forwarding attacks. An attractive approach to

decrease the impact of selective-forwarding attacks is to send each message through multiple paths to the sink. However, Fig. 3 shows that the improvement from multi-path routing is limited. In our simulations, the performance improvements were limited to 5% to 10% when two paths per message were used (one FTree path and another RRobin path), for different resiliency levels. In both RESIST-1 and the vulnerable cases, using more than two paths would not be sufficient to reach the performance of RESIST-0.

C. Routing Protocols Comparison

We next focus on comparing the performance of the three routing protocols using RESIST-0 (see Fig. 4(a)). Note that even if sinkhole attacks are avoided, malicious nodes can still perform selective forwarding attacks. Fig. 4(a) clearly shows that *FTree* and *RRobin* outperform *RWalk*. This is expected as in *RWalk*, the average path length that each message travels to the sink is longer. This consequently increases the probability of meeting a malicious node on the path. Further experimentation on *RWalk* also showed that the protocol performance is inversely proportional to n. This actually means that the best case for n-hop random walk is achieved when $n = 0$, in which case *RWalk* is equivalent to *FTree* routing.

Fig. 4(a) also shows that *FTree* and *RRobin* have always comparable performance regardless of the risk factor. This is surprising since, intuitively, the performance of *RRobin* should be better compared to *FTree*. Analyzing the results, we observe that, as expected, for sensors, which have malicious parents, *RRobin* improves the performance by letting these nodes periodically send to alternative parents. However, this does not necessarily improve overall performance as *the reverse case* also holds: sensor nodes with good parents on the routing tree use malicious nodes as parents in a round robin fashion. Consequently, any gain from *RRobin* is neutralized by putting sensor nodes with good parents at risk.

To understand the effect of malicious nodes on the protocol behavior better, we divide the network into 100 equal zones and define the *failure threshold of a zone* as the percentage of data sent by the zone that needs to be dropped to qualify the zone as poorly monitored. In reality, this threshold would depend on the criticality of the sensor network application. We set the failure threshold as 60% in our experiments. Fig. 4(b) illustrates how many zones fell below the failure threshold
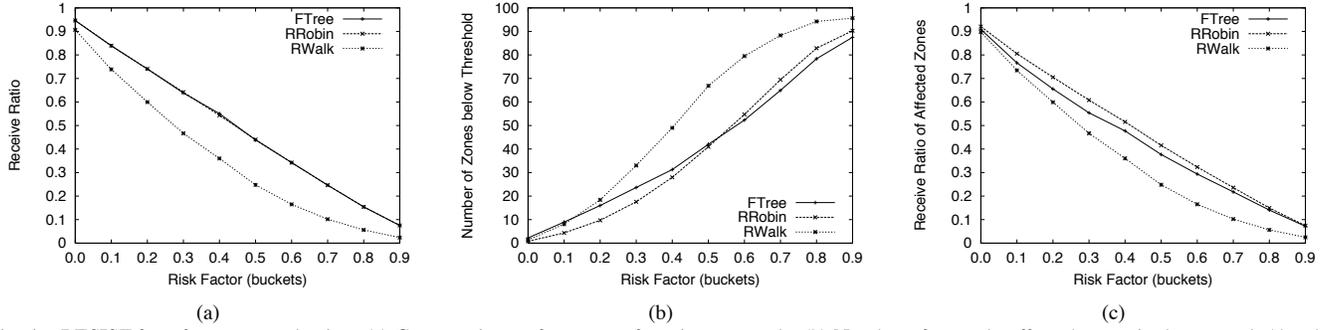
Fig. 4. RESIST-0 performance evaluation: (a) Comparative performance of routing protocols. (b) Number of severely affected zones in the network (threshold =60%). (c) Received ratio of affected zones in the network.
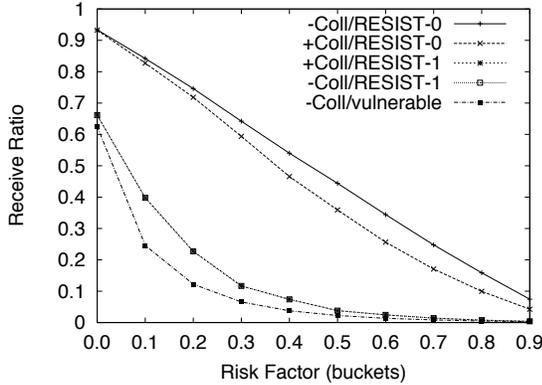

Fig. 5. In the presence of short range collusion (+Coll), RESIST-0 still performs much better than vulnerable routing without collusion. RESIST-1 is not affected, since the default malicious behavior is similar to collusion.

for each risk factor bucket and routing protocol. Initially, the number of zones below failure threshold is higher for *FTree* than *RRobin*. Coupled with the fact that both protocols share the same receive ratio (see Fig. 4(c)), this means that *RRobin* just diffuses the effect of malicious nodes to more zones, so that fewer zones actually fail. However, as the risk factor increases, the number of zones below threshold increases beyond *FTree* due to *the reverse case* appearing more often. Essentially, increasing the failure threshold moves the shift point to the right. Nevertheless, although the number of affected zones is higher for *RRobin*, the average received data ratio per affected zone still remains higher than *FTree*. On the other hand, the number of failed zones in *RWalk* is always the highest due to its overall poor performance.

V. DISCUSSION

We discuss here the cost of using cryptographic primitives in WSNs and the impact of collusion on RESIST protocols.

a) Reducing the Cost of Cryptography for Sensors: The main cost of RESIST is the size of `Hello` messages, which carry multiple tokens that contain cryptographic values. Additionally, although RESIST-1 does not introduce any overhead as it piggybacks to tree generation messages, RESIST-0 also requires two additional messages, `Challenge` and `ChallengeReply` to provide stronger resistance. However, while this overhead of RESIST-0 cannot be immediately

reduced, we believe that the cost of `Hello` messages can be significantly decreased by using an efficient way to implement tokens. First, Elliptic Curves Cryptography (ECC) keys can be used to reduce the size of messages. Signatures and keys of length 110 bits would be sufficient, since ECC keys are as efficient as much longer RSA keys. Recently, [13] shows that ECC can be implemented at a very low cost in WSN.

Second, similar to [16], one-way hash functions can be used to reduce the overhead of RESIST. In RESIST-1, a token T_{k+1} can be generated from a token T_k. Hence, `Hello` messages need only to contain the first and the last token, signed by the sink, i.e. $\texttt{Hello}(H^k(T_0), (epoch, H^D(T_0))_{signed(K_{pri}^{sink})})$, where D is the diameter of the network. When a sensor at a distance k receives the `Hello` message, it needs to hash the token $H^k(T_0)$ until it reaches $H^D(T_0)$. The number of times it needs to hash, x, defines its distance from the sink $(k = D - x)$. Moreover, as the sensor does not know T_0, and H is a one-way function, it cannot compute $H^{k-1}(T_0)$ and claim a shorter distance.

Similarly in RESIST-0, the public keys K_{pub}^k and K_{pub}^D need only be propagated. Then, the sensors use K_{pub}^k to verify the challenge. They also use it to generate the next hop key (implemented using ECC) K_{pri}^{k+1} (with the cost of a modulo operation), from which the public key K_{pub}^{k+1} is then generated (with the cost of an exponentiation operation). This new key is then sent to the neighbors. The chain can easily be verified by iterating between the two operations until K_{pub}^D is reached.

b) The Impact of Collusion on RESIST Protocols: To be able to implement sinkhole attacks in the presence of RESIST, malicious sensors need to collude to share good tokens (i.e., a token that can prove a short distance to the sink) they are able to collect during reconfigurations. Thus, RESIST protocols limit the power of collusion attacks to the closest distance an attacker can get to the sink. Hence, malicious nodes residing at the border of the network, as in our initial scenario in the introduction, would not be able to disrupt it.

Collusion is also limited by the communication capabilities of malicious sensors: in Fig. 5, we simulated the impact of collusion when colluding sensors and are distributed randomly on the network area. In our simulations, malicious nodes exchange tokens so that they all appear the same distance to the sink (i.e., the distance of the malicious node that is closest to the sink). Our results show that the performance of

RESIST-1 is not affected by the presence of colluding nodes. This was expected, as the collusion among malicious nodes do not necessarily create a higher impact on the structure of the tree. On the other hand, in RESIST-0, sharing of tokens enables replying challenges for shorter distances and hence, has an effect on performance.

Finally, the most dreadful attack would be a malicious sensor, close to the sink with a long radio range, propagating good tokens to malicious sensors far from the sink. However, collusion in this case might not cause a significantly higher degradation in performance, as the dominant impact already comes from the malicious node closest to the sink.

VI. RELATED WORK

Security has been attracting the attention of many researchers [17], since it is vital to guarantee correct operation of sensor protocols. The main conclusion of recent studies on secure routing is that updating current protocols with security extensions is not sufficient and that routing protocols should be designed from scratch with security in mind.

This paper focuses particularly on sinkhole and selective forwarding attacks. Most other approaches against these attacks revolves around detection of malicious nodes [1], [2], [16]. Multi-hop acknowledgments are used in [2], to detect and blacklist nodes that perform selective forwarding attacks. However, in addition to its cost, the proposed scheme requires geographical location information and strict synchronization. In [1], a learning technique based on neural networks is used to predict the sensor measurements, and a reputation scheme is used to mark nodes as faulty if their reports are too different from predictions. In [16], a protocol similar to RESIST-1 is proposed, but without strong cryptography. As a consequence, it requires an additional protocol to detect malicious sensors (reports are vulnerable to falsification) and to blacklist nodes (through a complex messaging mechanism).

An interesting analysis of DDoS attacks in sensor networks, which also takes into account different network parameters and some counter measures, is presented in [18]. While this work covers TCP JellyFish and selective-forwarding attacks, we focus on sinkhole attacks. Moreover, our Risk Factor metric is able to capture more network characteristics.

An intuitive approach against selective forwarding attacks is to use multipath routing [19]. However, such a protocol dramatically increases communication overhead as the redundancy of paths increases. In addition, these paths eventually converge to a few nodes surrounding the base station where malicious nodes can have a dreadful impact. Indeed, our simulation results show that the efficiency of this approach is limited, as confirmed by [18].

Trust-based systems [20] are interesting approaches to deal with selective forwarding attacks. In these systems, interactions between sensors are used for trust level computation. Such systems are, however, often complex. We believe resilience, as provided by our protocols, is a better choice. As in [21], RESIST could use trust levels in the choice of the set of nodes to be considered during the round robin procedure.

VII. CONCLUSION

We presented RESIST, which increases resilience to sinkhole attacks in WSNs: RESIST-1 prevents malicious nodes from changing their advertised distance to the sink more than one hop; RESIST-0 completely stops any lying about distance, but is more expensive to use. Our evaluations show that one indeed needs to pay this price, as RESIST-0 achieves significantly better performance for all risk factors, while RESIST-1 provides resistance over vulnerable case for low risk. Furthermore, different routing protocols are shown to have different effects on performance. For instance, if protocols increase path lengths to achieve path diversity (i.e., RWalk), this only increases failure probability. Additionally, just using randomness to escape from malicious nodes (RRobin) is not enough and it is necessary to check that good paths are not left for the bad paths. Finally, collusion between compromised sensors, even if possible, has limited impact on the real performance of our protocols.

REFERENCES

[1] P. Mukherjee and S. Sen, "Using learned data patterns to detect malicious nodes in sensor networks," in *ICDCN*, 2008.
[2] B. Yu and B. Xao, "Detecting selective forwarding attacks in wireless sensor networks," in *IEEE IPDPS*, 2006.
[3] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Sensys*, Nov. 2005.
[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *ACM MOBICOM*, 2000.
[5] F. Ye, A. Chen, S. Lu, and L. Zhang, "Gradient broadcast: A robust, long-live large sensor network," UCLA, Tech. Rep., 2001.
[6] ——, "A scalable solution to minimum cost forwarding in large sensor networks," in *Conf. on Computer Communications and Networks*, 2001.
[7] U. Cetintemel, A. Flinders, and Y. Sun, "Power-efficient data dissemination in wireless sensor networks," in *ACM MobiDE*, 2003.
[8] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *IEEE ICDCS*, 2002.
[9] Y. J. Zhao, R. Govindan, and D. Estrin, "Residual energy scans for monitoring wireless sensor networks," in *IEEE WCNC*, 2002.
[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks," *OSR*, 2002.
[11] L. B. Oliveira, D. Aranha, E. Morais, F. Daguano, J. Lpez, and R. Dahab, "Identity-based encryption for sensor networks," in *PERCOM*, 2007.
[12] S. A. Çamtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," *IEEE/ACM Trans. on Networking*, 2005.
[13] D. J. Malan, M. Welsh, and M. D. Smith, "Implementing public-key infrastructure for sensor networks," *ACM Trans. Sen. Netw.*, 2008.
[14] A. Papadimitriou, F. Le Fessant, A. C. Viana, and C. Sengul, "Fighting sinkhole attacks in tree-based routing topologies," INRIA, Tech. Rep. RR-6811, 2008.
[15] F. Le Fessant, C. Sengul, and A.-M. Kermarrec, "Pace-maker: Tracking peer availability in large networks," INRIA, Tech. Rep. RR-6594, 2008.
[16] S.-B. Lee and Y.-H. Choi, "A secure alternate path routing in sensor networks," *Computer Communications, Elsevier*, vol. 30, 2006.
[17] W. Yu and K. Liu, "Attack-resistant cooperation stimulation in autonomous ad hoc networks," *IEEE/ACM Trans. on Networking*, 2005.
[18] I. Aad, J.-P. Hubaux, and E. W. Knightly, "Impact of denial of service attacks on ad hoc networks," *IEEE/ACM Trans. on Networking*, 2008.
[19] S. M. Jing Deng, Richard Han, "Insens: Intrusion-tolerant routing in wireless sensor networks," in *IEEE ICDCS*, 2003.
[20] Y. Sun, Z. Han, W. Yu, and K. J. R. Liu, "A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks," in *IEEE INFOCOM*, 2007.
[21] Z. Liu, A. Joy, and R. Thompson, "A dynamic trust model for mobile ad hoc networks," in *Future Trends of Dist. Computing Systems*, 2004.