Advanced Evasive Data Storage in Sensor Networks

Zinaida Benenson Felix C. Freiling
University of Mannheim, Germany

Peter M. Cholewinski
SAP - Research and Breakthrough Innovation, Germany

***Abstract*—In case the data which is stored and processed in a sensor network has some value, it needs to be protected from unauthorized access through a security mechanism. The idea of evasive data storage is that data moves around the sensor network instead of remaining at a fixed location. In this way, an adversary, who has once (through node capture) had access to the data stored at some particular node, must compromise more sensors in order to maintain his illegitimate access to the sensor data. We refine the previously published Simple Evasive Data Storage techniques in two ways: (1) we improve the efficiency of data retrieval by bounding the area in which data may move, (2) we introduce Data Splitting as a technique to protect against *sleeper attacks* in which the adversary simply takes over a subset of nodes and waits for valuable data to pass by. We demonstrate the effectiveness of our approach using extensive simulations.**

I. Introduction

Wireless sensor networks are networks of tiny computing devices that are equipped with sensors to measure environmental conditions like temperature, humidity, movement or noise. The sensors are usually also equipped with a wireless communication device to communicate with other sensors. Applications of sensor networks range ¿from environmental monitoring (fire control, seismic activity) over process control (monitoring of physical/chemical production processes) to military applications (location and battlefield surveillance).

Sensor networks can be regarded as widely distributed databases which store and process sensed data. In case the data which is stored the sensor network has some value, it needs to be protected from unauthorized access through a security mechanism. One of the most obvious ways to illegally access the data in a sensor network also has one of the most obvious solutions: An attacker which eavesdrops on the data that is transmitted over the wireless communication link can easily be defeated by encrypting this data. Standard low-cost mechanisms exist to establish pairwise encrypted channels in sensor networks [10], [15], [5], [8]. Other attacks however are not so easily defeated.

One of the most dangerous attacks on sensor networks is *node capture*. A node capture occurs if an adversary completely takes over a sensor node and uses it to spy on valuable data which is stored and processed within the sensor and the entire sensor network. Node capture is a realistic threat since sensor nodes usually do not have any mechanism of tamper-resistance and accessing their internals (memory, secret keys, etc.) is relatively easy. Node capture is also hard to detect and so remains "one of the most vexing problems in sensor network security" [9].

Different concepts have been introduced to withstand node capture attacks in sensor networks. Most of them exploit the massive spatial redundancy in sensor networks. Vogt [14] focusses on message authentication and uses multiple authentication paths to detect corrupted nodes. In a similar direction, Przydatek *et al.* [11] investigate secure data aggregation by using random sampling and interactive proofs to eliminate cheaters. Karlof and Wagner [7] focus on secure routing. A different paradigm to combat node capture was proposed by Bicakci *et al.* [3]. Their *one-time sensors* can only be used a limited number of times and therefore quickly become useless to an attacker. We are aware of only one paper which considers secure storage in sensor networks: Ghose *et al.* [6] investigate the resilience against loss of certain nodes in data-centric storage by distributing information over several nodes. In this paper we follow the distinct approach of Evasive Data Storage (EDS) which was presented in previous work [1].

EDS in sensor networks was introduced to limit the power of an adversary which is able to perform node capture. The basic idea of EDS is that valuable data does not reside in a fixed location in the network, but wanders around the network in an unpredictable fashion. An attacker, who has captured a node storing aggregated (and therefore precious) data will at some point in time lose access to this data since the data will not be stored in the captured node anymore. EDS therefore goes one step beyond methods like Data Centric Storage [13] or Local Storage (each node stores the data generated by it). EDS can therefore drastically improve the security of a sensor network if the right strategy of evasion is chosen [1].

Two concerns about the simple EDS schemes in previous work have been raised. The first concern deals with the fact that existing simple EDS algorithms are not very communication efficient. Especially retrieval operations necessarily need to flood the network with requests since the maintainers of the sensor network have no more knowledge about the whereabouts of the data than the attacker. The second concern deals with the problem of so-called *sleeper attacks*. In a sleeper attack the adversary simply takes over a subset of nodes and waits for valuable data to pass by.

In this paper we present more advanced variations of the EDS notion which avoid the above problems. The first enhancement improves the retrieval properties of the simple algorithm. The Location Bound EDS approach restricts the area where evasion may occur. This generalization basically introduces a parameter that connects the two extremes of

Zinaida Benenson was supported by Landesstiftung Baden Württemberg as part of Project "ZeuS".

1-4244-1241-2/07/$25.00 ©2007 IEEE

traditional storage and evasive storage, therefore granting more control over the process and a decrease in the complexity of the data retrieval.

The second enhancement is Data Splitting (DS) which can successfully counter sleeper attacks. The idea is to split data in smaller pieces which will then wander through the sensor network independently. This basically means that the infiltrator looses his jackpot advantage: without DS a malicious node will win the instance data is moved to it; with splitting however, malicious nodes have to collect a bunch of small pieces until they can reassemble the complete data record.

In summary, EDS is another instance of the "guerilla tactics" approach to security in sensor networks [2].

We first present some definitions and the model of a sensor network we use in the algorithms in Section II. Then we recall SEDS [1] in Section III. Finally, we present the concepts of Location Bound EDS and DS in Sections IV and V, respectively. For lack of space we only give algorithm and proof sketches here. The full details as well as the source code of the simulations can be found elsewhere [4].

II. Definitions and Model

A. Network Model

Considering a snapshot of a sensor network at a given instance, we assume n nodes in total, among which h *hot* nodes can be found. Hot nodes are assumed to store precious data and are the target of attackers.

Following common notation for algorithms in sensor networks, we also present the steps from the point of view of a single sensor node, denoted as s. This node can only interact directly with nodes in its vicinity $V(s)$, where each one such node can be referred to as a vicinity node $v \in V(s)$. The vicinity of a node is mainly defined by the strength of its wireless communication device it disposes of.

If it is necessary to refer to a message sent in the network without giving away any detail about its contents or structure the notation msg is used to identify such a message. We assume that communication links are encrypted.

B. Communication Primitives

Nodes communicate using *local broadcast*, i.e. a node s can communicate with all nodes $v \in V(s)$ with a single message in one communication step. Local broadcast is assumed to be probabilistic, i.e. the probability that a node in $V(s)$ receives the message is p_{lb}. This probability also influences other types of communication which are build on top of local broadcast, including *point-to-point* message passing and *global broadcast* primitives. The probability p_{lb} can be ascribed to the harsh environment the nodes exposed to (packet loss), as well as to the possibility of failure.

C. Adversary Model

We define several different models for the adversary that reflect realistic possibilities. An adversary $\mathcal{A}$ can exhibit two basic skills, namely *traffic analysis* and *intervention* skills. The former can take the levels *blind*, *local* and *global*, whereas the latter is subdivided into *passive* and *active*. The traffic analysis levels merely describe the region an adversary can observe, rendering a blind adversary inapt to perform any traffic analysis, allowing a local adversary to observe a fraction of the network and ascribing the global adversary observability of the complete network. Although the local adversary might call for a more detailed definition of the boundaries imposed on the local region he can observe, for our purposes it suffices to state that he cannot track traffic in the complete network (in most cases it can be assumed that he can observe traffic generated by nodes that are in his vicinity which is only little larger than that of an average sensor node).

In the case of *intervention*, the levels describe how much power an adversary has to influence the behavior of sensor nodes: the passive case allows an adversary only to extract data from a node, whereas the active case allows for complete take over of the node including the scenario where the node is forced to exhibit arbitrary behavior.

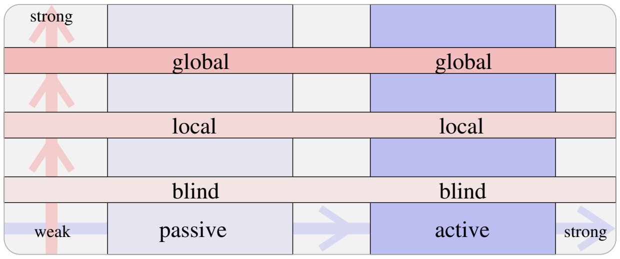

Fig. 1. Adversary Lattice

Those two basic skills, traffic analysis and intervention, are orthogonal and define a lattice of different types of adversaries, depicted in Figure 1. An adversary will always be specified as a pair of skills, where the first component indicates the traffic analysis level and the second the intervention level. For instance, an adversary $\mathcal{A}(local, passive)$ can observe traffic only in his vicinity and when accessing a node only data extraction is an option for him, disallowing him to influence the behavior of the node.

In this paper we want to stress the case of a passive adversary that tries to access specific data in the network. With non evasive storage methods such an adversary can access data easier than in the evasive case, especially when he is looking for updates of data he already accessed. This property should become clear from the description of the notion in the following section. As we are proposing advanced variations of the EDS, we are also going to step into the realm of an active adversary, when taking a look at the DS notion.

III. Simple Evasive Data Storage

We now recall the SEDS algorithm [1]. The idea behind EDS can be put in quite simple terms. Distinguishing between the possibility of fixing the h hot nodes and enabling those h nodes to vary over time, gives rise to data evasiveness. However this is different from the movement of data seen, for example, in Data Centric Storage [13], [12], where data is moved to other nodes for storage, because once data reaches that final storage node, the node will remain hot. Data evasiveness, on the other hand, is the process of intentional shifting of data as part of long term storage, i.e. hot nodes can loose their status of being hot as time passes by, all to achieve the ultimate goal to defend against illegitimate retrieval of data.

As an adversary approaches a node that by his knowledge used to store data at some past instance of time, the EDS algorithm will have moved data in several displacement steps away from that node, thus an adversary will not succeed with probability 1.0 as in conventional approaches, but will have to try out close neighbors of the former hot node with a drastically lowered success rate.

A. Algorithm

A simple algorithm that implements the basic notion of EDS makes a hot node s, that received or generated some data D, *actively* choose which node $v \in V(s)$ should become the new harbor for D.

More specifically, the following sequence shows what is performed by a hot node s:

- s sends an evasion request to all $v \in V(s)$
- all $v \in V(s)$ respond indicating participation
- s receives responds and invokes the $Choice$ choice function on the nodes that indicated willingness to participate
- $Choice$ returns a single node, the chosen node c
- s transmits the data D to c
- c acknowledges reception

Naturally, transmission of the data D to the chosen node c needs to be encrypted such that other nodes in the vicinity cannot obtain data easily and also to prevent an adversary $\mathcal{A}$ to eavesdrop on the transmission gaining unauthorized access to data D without any substantial effort.

The algorithm now has to be put into context of storage: the assumption made is that data starts wandering directly after being generated, thus it is never assigned any particular node nor are means provided to inform base stations or other entities of its existence. Hence, to complete the approach to form a data storage mechanism, a short discussion of data retrieval is necessary. As there is no knowledge available concerning the whereabouts of data in the network at the time a legitimate user queries the network for data, the most straightforward approach for retrieval is *flooding*.

Compared to conventional storage the simple evasive data storage algorithm naturally lowers the success probability of an adversary $\mathcal{A}$ that tries to access a hot node periodically for updated data items: in conventional storage his chances are (almost always) equal to 1.0, whereas in EDS his chances are substantially less than 1.0. Furthermore a more detailed evaluation regarding how much worse the chances of $\mathcal{A}$ are can only be obtained when more information about the employed choice function $Choice$ is available.

B. Choice Functions

The core of the EDS notion is the parameter $Choice$, i.e. the concept of a Choice Function. Four fundamental choice functions and a specific combination technique have been proposed that naturally allow to obtain a myriad of choice functions. The four choice functions are given from the perspective of executing node s:

- UVicinity: choose $a \in V(s)$ with uniform probability.
- UFurthest$k$: choose $a \in V(s)$ with uniform probability among k furthest nodes from s.
- GNodeFurthest: choose $a \in V(s)$ that is furthest from the *generating node*, which is assumed to be provided by the respective data.[1]
- DirectionV: use an initial direction vector or change the current one with probability p_c to choose a node in $V(s)$ that is closest to the direction vector.

Of course, there are even further choice functions that can found and be argued for usability in different contexts, but in scope of this work we limit our attention to those just given and a simple combination technique, which resembles concatenation: given two choice functions $Choice_1$ and $Choice_2$, a new choice function $Choice^* = (Choice_1, e, Choice_2)$ results from making the first e evasion steps using $Choice_1$ and thereafter using the $Choice_2$ indefinitely. Hence, the new choice function $Choice^*$ can be seen as the concatenation of two choice functions, where the exact moment of changing from one function to the other is parameterized by e.

From a security point of view, it is most desirable to have the choice function form a uniform probability over the complete network, i.e., the probability of storing a particular data item is equal for all the nodes in the network. Previous work [1] identified a combination of two basic choice functions that provides satisfying results: (DirectionV, $\frac{e}{2}$, UFurthest$k$). This combination has revealed to have a desirable probability distribution, assuming the parameter $\frac{e}{2}$ is chosen adequately (taking into account the size of the sensor). To put in words what this function basically does: it will first move the data away from the original node in some random direction and then evade the data in the newly reached region.

C. Simulation Setup

As means of showing the effectiveness and workings of the proposed algorithms, we make use of simulations just as it was done in [1] for SEDS.

To investigate the EDS notions a network of 10000 nodes is set up within a rectangular grid and data is evaded for a certain number e of evasive steps starting some node (mostly located at $(0, 0)$). Also the simulations were carried out with a vicinity radius of 3, thus strongly limiting the amount of reachable neighbors of nodes as otherwise the algorithms and choice functions could not have been evaluated properly. As most of the algorithms offer quite some parametrization possibilities (for instance regarding triggering intervals), we will mention only the crucial values used when referring to the simulation results. Also if one of the basic parameter is changed, we will point that out in the text.

[1] Other means to specify the *generating node* are also possible, even ones that do not correspond to the genuine intention of this choice function.

The crucial goal to be achieved in most of the simulations is to show that an adversary is bestowed with a lower success probability for data retrieval when one of the EDS algorithms is used.

IV. Location Bound Evasive Data Storage

This section aims to present a refinement to the SEDS algorithm yielding better properties when performing data retrieval and update respectively, but not loosing the security advantages inherent to EDS at the same time. The notion introduced will be referred to as Location Bound EDS.

Unlike the conventional assumptions that the sensor network contains data stored in h distinct hot nodes (which remain hot over their life time), the Location Bound EDS allows for h distinct hot regions that are responsible for storing the data gathered by the sensor network's nodes. Before completely diving into the notion of Location Bound EDS the concept of a hot region or *hot group*, as it will be referred to, needs to be introduced.

A hot group H is a set consisting of d sensor nodes such that for each arbitrary pair $a \in H$ and $a' \in H$, there is a way to exchange messages by using nodes within H only. Thus, nodes in a group can move data around without having to rely on nodes that are not members of their respective group.

Now that hot groups are a familiar term, Location Bound EDS can be talked about in more detail. It forms an extension to the traditional notion of having h hot nodes in a sensor network by replacing single hot nodes with h distinct hot groups, where each group has at most $d \geq 1$ members in order to store the gathered data. It is tried to keep the cardinality of each H to be as close to d as possible at all times, however due to failures this cannot be guaranteed at every point in time. Within each such group H data is assumed to be able to move around freely (an evasive storage mechanism is employed), but is not allowed to leave the group, therefore restricting the impact evasiveness has on the overall traffic within the network, but more importantly: restricting the location where data needs to be looked for.

The definition of Location Bound EDS has the property of fitting nicely into the picture that has been presented up till now:

- ○ setting d to be equal to 1, the Location Bound EDS becomes the now widely assumed storage where data is fixed at h distinct hot nodes.
- ○ at the other extreme, setting $d = n$, hence making the whole network available to evasiveness, the concept collapses to the SEDS algorithm introduced in the last section. [2]

Note that there is no restriction in the definition of hot groups that disallows the possibility of having some node a belong to two distinct hot groups. This is the reason why setting $d = n$ works out to be possible and results in the SEDS algorithm from the last section. In most cases however, the setup of hot groups will avoid multiple membership of nodes.

Having given the definition, all we need to make clear now is that employing SEDS within those hot groups will allow us to attain a uniform probability distribution within the hot region by using (DirectionV, $\frac{e}{2}$, UFurthest$k$) as Choice Function. Figure 2 shows the probability distribution that results after $e = 18$ steps originating from $(0, 0)$ (middle of the plane shown in the figure). What happens therefore, is that the first nine steps are used to move data away in a random direction and the remaining ones more data in the reached region.

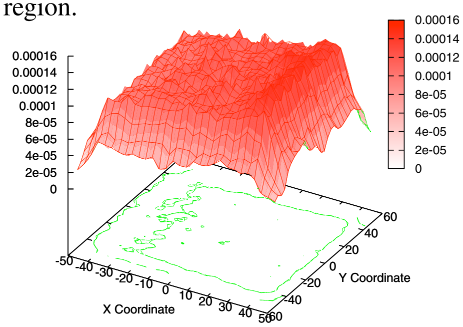

Fig. 2. Probability distribution after 18 evasive steps for choice function (DirectionV, e,UFurthest$k$).

Thus the probability distribution shows what we have gained: an adversary has to consider all nodes in a specific region in order to obtain data, which given an adequate size of a hot group can still lead to significant security properties[3]. One the other hand, as we have restricted the area, we have alleviated the SEDS notion from its inherent data retrieval problems: now no flooding of the complete network is necessary anymore, but can be restricted to the comparably smaller region of a hot group.

We have also conducted simulations with implementations of the Location Bound EDS and show a simulative picture of the algorithm in Figure 3 for a 18 evasive steps (grey marks the hot groups in the network). The Choice Function used here for Location Bound EDS is DirectionV with $p_c = 0.05$ implying that a direction change will occur with a small probability during an evasion step.

Figure 3 also points out some details of the approach that need to be mentioned. The hot groups have to be setup somehow. The simplest way to achieve this is use the same technique as done in Data Centric Storage: have a global hash table deployed at each node, which then has pre-configured locations assigned to hot groups. Hence, every node will be able to identify which hot group it should send its data for (evasive) storage to and also have knowledge about the whereabouts of the group.

Thus as shown in the figure, three data items are assigned to three hot groups located in the network and forwarded directly

[2] This presumes that the evasiveness approach used in a hot group is actually equal to SEDS.

[3] Now the e parameter has to consider the size of the hot group instead of the whole network.

to a head node of the group, from which the evasive storage is initiated. It should be clear that hot groups ought to fulfill two properties: (1) the union of all hot group member nodes should be significantly smaller than n, and (2) intersections of hot groups should be avoided. The latter is especially important when considering energy consumption, as participating in several hot groups will put a higher energy drain on a node.

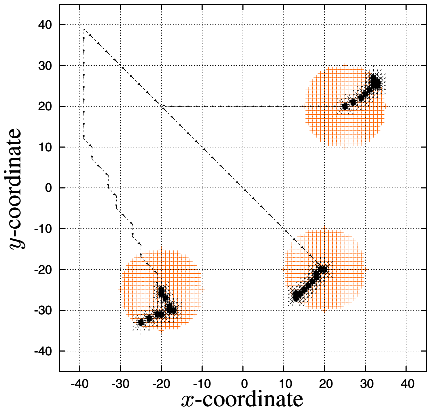

Fig. 3. Location of hot groups after 18 evasive steps for choice function DirectionV with $p_c = 0.05$

V. Evasive Data Storage with Data Splitting

A crucial (security) problem that can be ascribed to simple EDS is the possibility that an adversary can infiltrate the network with malicious nodes, which we know from the adversary models to be an active adversary $\mathcal{A}$. The malicious nodes $\mathcal{A}$ takes over can idle until data is evaded to them, hence making it even simpler for $\mathcal{A}$ to get data. Of course, this scenario needs to be put into context: a realistic number of malicious nodes is minute, thus we felt that the problem is not as severe as to render the whole approach less useful, but still should deserve some attention. A simple solution is offered by the DS notion, which splits data D into s many splints that contain only a fraction of the information actually stored in D. The complete description of the generation of those splints includes the usage of the assumed cryptographic primitive with the goal in mind to encrypt D before splitting it up, such that every single splint does not provide any significant information to an adversary until he collects all splints of D. Of course, the usage of cryptographic primitives has to consider the restrictions imposed by sensor networks, thus also forces usage of simple methods only for DS.

One very simple method would be to create some permutation P for data D, which then acts on D to obtain D'. Encrypted D' is then divided into splints of equal size $d_1, \dots, d_s$; where for storage one is attached the necessary information for reversing P. Of course, it is assumed that P can only be used for decryption of data when all s many splints are available. Otherwise, $\mathcal{A}$ could get lucky when he catches the one splint including P-appendages and is able to extract some information.

How does DS increase security given that malicious nodes wait for data to be displaced into their palms? This question is easily answered assuming that a single splint does not provide an adversary $\mathcal{A}$ with any significant information about the data D at hand. Then in order for $\mathcal{A}$ to be successful, he needs to collect all s splints before capturing D, which naturally will not happen as easily compared to the case where no splitting is applied. The main reason for this is that each splint can take a different path in the network during evasion. Hence, we can also identify the parameters that influence the success of DS which are the size of the network as well as the value of s. The former is providing the splints with sufficient space not affected by malicious nodes to explore. The s increases the work for the adversary, as a larger value for s implies a longer time interval until $\mathcal{A}$ can collect all splints. We will provide more accurate information about the usability of DS in the discussion of the simulation results to follow.

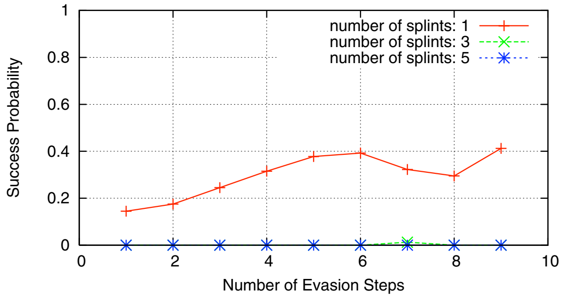

Fig. 4. DirectionV ($p_c = 0.05$) with non-cooperative 5 percent malicious nodes

The main parameter that influences the properties of the algorithm is s, the number of splints that the splitting is supposed to generate. The steps algorithm performs are simple: instead of storing the data item D with an EDS algorithm, the data item is preprocessed, split, and then each of the resulting splints is passed on for storage with one of the EDS, like Location Bound EDS from the last section. We will not dive into too much details of how the actual splitting is done or what factors are necessary to take into account (encryption, for instance). What we want to comment on are the simulation results that are shown in Figures 4, 5, and 6 respectively. The simulations are performed in a network of 3600 nodes, i.e. a place of 60x60. The evasion is started at location $(0, 0)$ and malicious nodes are randomly distributed in the network.

Figure 4 shows the success probability an adversary that infiltrated 5 per cent of the nodes in a sensor network in dependence of the number of steps. Here the simulative results consider non-cooperative malicious nodes. The non-cooperative property states that every malicious node has to collect all splints itself, before it can extract any data from its collection. It can be seen that without DS the adversary can achieve remarkable results from his sleeper attack, however even a very small splitting size s brings significant improvement.

Even when considering collaborative malicious nodes as

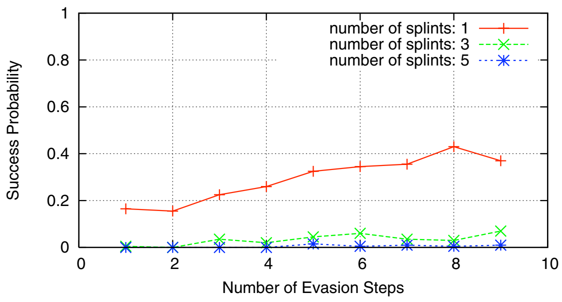

Fig. 5. DirectionV ($p_c = 0.05$) with cooperative 5 percent malicious nodes

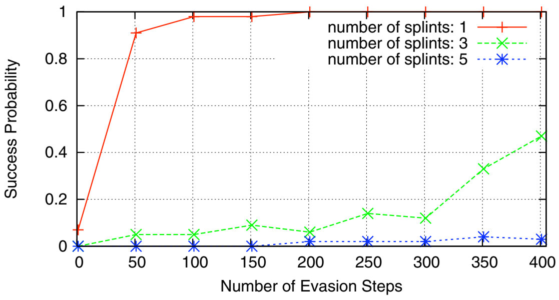

Fig. 6. Long term behavior with non-cooperative 5 percent malicious nodes

shown in Figure 5, the impact on DS is not heavy. Although the malicious nodes are now assumed to be able to exchange all the collected pieces in order to puzzle together the data, their gain from doing so is minute. As can be seen for $s = 3$, the probability increases only insignificantly. Nonetheless, the rise of the success probability with the number of evasive steps, asks for a look at the long term behavior.

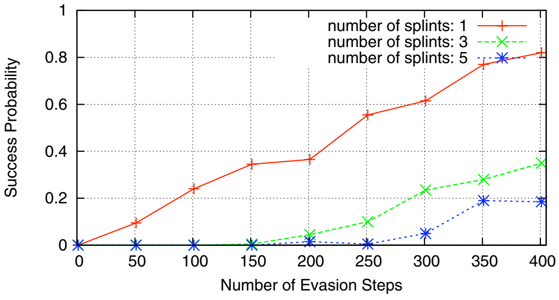

Fig. 7. Long term behavior with cooperative .25 percent malicious nodes

Hence, we will see what happens after several hundred evasion steps, which is considered to span over a very long period of time during the sensor networks life-period. Figure 6 shows results that consider DS with EDS under a non-cooperative sleeper attack. It becomes clear that DS is only a technique that buys time, but will not allow to counteract sleeper attacks indefinitely. We have omitted the 5 percent figure for the cooperative case, as this showed that such a large cooperative sleeper attack renders DS not very effective for the considered s values of three and five (the curves are almost identical to the curve of EDS without DS).

Figure 7 indicates that under more realistic assumptions of a low number of hostile, cooperative nodes in the network, the addition of DS does indeed help to counteract successful data retrieval, even for small amounts of splints. But again, only for a bounded time-span: eventually all curves will converge for arbitrary s values. We do not see this as a problem, as a large number of evasion steps needs to take place until a convergence is reached and this would certainly correspond of a time span well beyond the lifetime of a sensor network.

REFERENCES

[1] Z. Benenson, F. Freiling, and P. Cholewinski. Simple evasive data storage. In *Proc. 17th IASTED International Conference on Parallel and Distributed Computing and Systems: First International Workshop on Distributed Algorithms and Applications for Wireless and Mobile Systems*, 2005.

[2] Z. Benenson and F. C. Freiling. On the feasibility and meaning of security in sensor networks. In *4th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze". Tech. Rep. TR 481, Department Informatik, ETH Zurich*, Zurich, Switzerland, Mar. 2005. http://www.vs.inf.ethz.ch/events/fg2005/fgsn05.pdf.

[3] K. Bicakci, C. Gamage, B. Crispo, and A. Tanenbaum. One-time sensors: A novel concept to mitigate node-capture attacks. In *2nd European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS2005)*, 2005.

[4] P. M. Cholewinski. Evasive data storage in sensor networks. Diploma thesis, RWTH Aachen University, Germany, Department of Computer Science, 2005. Simulation code can be downloaded at https://pi1.informatik.uni-mannheim.de:8443/pub/research/downloads/EDS-Simulator-1.0.zip.

[5] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. 9th ACM conference on Computer and Communications Security*, pages 41–47. ACM Press, 2002.

[6] A. Ghose, J. Grossklags, and J. Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *MDM '03: Proceedings of the 4th International Conference on Mobile Data Management*, pages 45–62. Springer-Verlag, 2003.

[7] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Network Journal*, September 2003.

[8] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proc. 10th ACM conference on Computer and Communications Security*, pages 52–61. ACM Press, 2003.

[9] A. Perrig, J. A. Stankovic, and D. Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.

[10] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. Tygar. SPINS: Security protocols for sensor networks. In *Proc. MOBICOM*, 2001.

[11] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *ACM SenSys 2003*, Nov 2003.

[12] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage in sensornets. In *Proc. First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, Sept. 2002.

[13] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. In *Proc. First ACM SIGCOMM Workshop on Hot Topics in Networks*, Oct. 2002.

[14] H. Vogt. Exploring message authentication in sensor networks. In *Security in Ad-hoc and Sensor Networks (ESAS), First European Workshop*, volume 3313 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 2004.

[15] S. Zhu, S. Setia, and S. Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proc. 10th ACM conference on Computer and Communication Security*, pages 62–72. ACM Press, 2003.