Proceedings of the 17th IASTED International Conference
Parallel and Distributed Computing and Systems
November 14-16, 2005, Phoenix, AZ, USA

SIMPLE EVASIVE DATA STORAGE IN SENSOR NETWORKS

Zinaida Benenson* Peter M. Cholewinski Felix C. Freiling†

ABSTRACT

We study a new countermeasure to the well known threat of *node capture* in sensor networks. A node capture occurs if an adversary completely takes over a sensor node and uses it to spy on the data which is stored and processed within the sensor network. In case this data has some value it needs to be protected from unauthorized access through a security mechanism. Merely encrypting the data is sufficient to protect its contents from eavesdroppers but not from node capture. We present a new approach, namely Evasive Data Storage, that intends to improve security features at the data storage level. 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. Through simulation we show that the new notion of Evasive Data Storage effectively decreases the probability that the adversary finds the data again even though he knew where the data once resided. Hence Evasive Data Storage offers an interesting possibility to secure data in sensor networks.

KEY WORDS

Sensor Networks, Evasive Data Storage, Camouflage, Node Capture Problem.

1 Introduction

Sensor networks offer a rich and challenging environment for the topic of data storage to be explored from different perspectives than it is done in conventional systems [8, 7]. The very idea that hostile adversaries can monitor traffic, access nodes, and thus gain access to data calls out for new solutions to protect the confidentiality of data in sensor networks.

Assuming that some precious data is stored at a subset of nodes in the sensor network, a security mechanism must ensure that an adversary cannot easily identify these nodes. The usual methods to achieve this are called *anti-traffic analysis* and use random network traffic to disguise the critical traffic which could be used to find the precious data. This idea is well-known from other areas of communications security and has already been employed in sensor networks [2].

Anti-traffic analysis is of no use if the adversary already has identified a critical node and has completely taken over control of that node. This action is commonly called *node capture* and is one of the most challenging adversarial behaviors which are possible in sensor networks. If a critical node is captured, the adversary is able to *always* access the data of the sensor network, no matter how much anti-traffic analysis overhead is employed.

In the presence of node captures, data cannot be sufficiently protected with the established conventional storage algorithms, because methods like Data Centric Storage [8] or Local Storage (each node stores the data generated by it) dictate that data is stored at a certain node and remains there for the lifetime of the network. We therefore propose the idea to replace the rigid dictation for data to remain at fixed locations to a more lax approach to storage that allows data to be wandering in the network. Allowing for such movement, we obviously can address the problem raised above, as malicious users cannot rely on the fact that they will find updated data at the same locations where it used to be previously. This shift in the data storage paradigm for sensor networks is the core of the Evasive Data Storage notion and the respective algorithms that implement the idea. The effectiveness of Evasive Data Storage depends on many parameters, for example the *evasion strategy* which defines the locations to which data may evade in the network.

Apart from introducing the notion of Evasive Data Storage we evaluate the effectiveness of different evasion techniques (which are encoded in a so-called *choice function*) for a simple Evasive Data Storage algorithm. By using simulations, we identify a choice function that makes it very hard to re-identify a critical node in the network after node capture. With the addition of anti-traffic analysis techniques as a security enhancing technique, we therefore provide an effective means to alleviate the severity of the node capture problem. These as well as more complex variants of Evasive Data Storage and anti-traffic analysis techniques are documented in a more detailed fashion in [1].

The paper is structured as follows: In Section 2 we introduce the model used for the algorithms and

*Work by Zinaida Benenson was supported as part of the Graduiertenkolleg "Software for mobile communication systems" funded by Deutsche Forschungsgemeinschaft.

†Contact author. All authors are affiliated with the Laboratory of Dependable Distributed Systems at RWTH Aachen University, Germany.

the simulations. In Section 3 we present the simple Evasive Data Storage algorithms and the simulation results of the different choice functions. We conclude in Section 4.

2 Definitions and Model

2.1 Network Model

We consider a network consisting of n nodes, where we distinguish between h *hot* nodes that store data at a certain instance, and $(n-h)$ *cold* nodes that never store data.

Most algorithms in sensor networks are presented from the point of view of a single sensor node, mostly denoted as s. This node can only interact directly with nodes in its vicinity $V(s)$, where one such node is 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. We assume that communication links are encrypted. Standard low-cost mechanisms exist to establish pairwise encrypted channels in sensor networks [6, 9, 3, 5].

2.2 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 built on top of local broadcast, including *point-to-point* message passing and *global broadcast* primitives.

2.3 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.

Those two basic skills, traffic analysis and intervention, are orthogonal and define a lattice of different types of adversaries. 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 an 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 following description of the notion.

3 Evasive Data Storage

We now explain the new notion of Evasive Data Storage. The goal of this new paradigm is to improve security properties of data given that a hostile entity is assumed to be located in the network's environment. Obviously, in order to increase security a price has to be paid and for the case of Evasive Data Storage better security is achieved through a certain message overhead that is imposed on the storage process. This also stresses the goal that this notion intends to achieve: providing protection against unauthorized data access with a high probability. Hence, we will not consider Denial of Service attacks that active adversaries might engage in, although we will comment on some malicious activities which are crucial to the effectiveness of Evasive Data Storage at the end of this paper.

3.1 Simple Algorithm

A simple algorithm that implements the basic notion of Evasive Data Storage 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. This view of evasive storage opposes the one where the node s solely initiates the displacement of D and delegates the actual decision to its neighboring nodes. Obviously, the latter view of evasive storage calls for consensus among the nodes, and also raises security issues regarding the transmission of data. The former view, the one we have decided to use for our

implementations of Evasive Data Storage, makes s the decision maker and does not lead to the problems just mentioned. To convey the simple algorithm in more detail, the following steps are 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 function *Choice* 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

This simple algorithm can be used to displace data items in the network. This 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 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*. Doing so, the legitimate user can cause a flooding of the network with a request for data and therefore eventually find the data wandering in the network.

Using the simple algorithm above, we can lower 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 Evasive Data Storage his chances are substantially less than 1.0. The exact estimation of the adversary's chances depend on the employed choice function *Choice*.

3.2 Choice Functions

The core of the Evasive Data Storage notion is the parameter *Choice*, i.e. the concept of a Choice Function. We propose four fundamental choice functions and a specific combination technique, that naturally allows 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.
- 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. The initial direction vector is chosen randomly and the probability that the direction changes is usually small (for the simulations we used $p_c = 0.005$).

Of course, there are even further choice functions that can be used 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.

3.3 Simulations

In order to estimate the security gain of Evasive Data Storage but also to stress the importance of the Choice Functions, simulations are employed. To do so two systems were used. The first, addressing the choice functions specifically, is a custom tailored environment written from scratch, the second used a publicly available discrete event simulator [4] in which all algorithms have been implemented in sufficient detail. The main objective to achieve here is to show that an adversary $\mathcal{A}$ needs to consider more nodes after having found a hot node and wanting to access the node for updated data compared to the conventional storage case. In the conventional storage case the same node will contain the data at any time during the network's life time. In Evasive Data Storage, many other nodes can be the new hot nodes, and the subsequent discussion shows that using the right choice function can force $\mathcal{A}$ to consider all nodes in the network (or a certain region) compared to only a single one in the conventional storage case.

To investigate the Evasive Data Storage a network of 10000 nodes is set up and data is evaded for a certain number e of evasive steps starting at node at location $(0, 0)$ using the Evasive Data Storage algorithm discussed above. In this algorithm, any particular choice function essentially leads to a probability distribution for the location of the data that wanders the network starting at $(0, 0)$. The figures which we will look at depict the probability that a node will become hot after e many displacements, which we can

also postulate to be equivalent to the success probability an adversary $\mathcal{A}$ has to pick a hot node knowing which choice function the storage mechanism uses. $\mathcal{A}$ then can be kept most effectively from accessing a hot node if the resulting distribution is uniform, i.e., all nodes have equal probability to become hot. We will show with the distributions yielded by the simulations that there actually is such a choice function that quickly yields such a desired uniform distribution. Note that the figures depicting the probability distributions for the choice functions have different z-ranges, which is necessary to guarantee interpretable images.

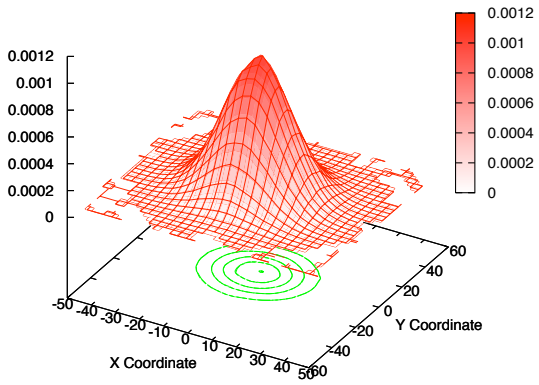

Figure 1. Evasive Data Storage with UVicinity after 6 evasive steps.

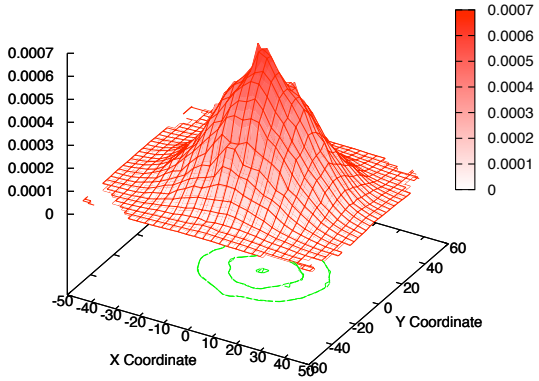

Figure 2. Evasive Data Storage with UFurthest$k$ after 6 evasive steps and $k = 2$.

Figure 1 depicts the probability distribution for the UVicinity choice function after $e = 6$ steps. Obviously, the probability for nodes to be hot after that amount of steps is quite high for nodes very close of the initial hot node located at $(0, 0)$. Thus, the $\mathcal{A}$ can easily obtain data by checking nodes close to $(0, 0)$. Similarly, the UFurthest$k$ exhibits a very similar distribution, though the peek around $(0, 0)$ is not as pointed as in the UVicinity case.

The remaining two choice functions GNodeFurthest and DirectionV are depicted in Figures 3 and 4 respectively. Here we observe that the former exhibits a form similar to a ring, see Figure 3, that propagates through the network centered at the original node, which is quite reasonable as always nodes that are furthest from position $(0, 0)$ are chosen. This might be an interesting property, however making e large enough the hot nodes will concentrate at the points furthest from the initial hot node, which can be exploited by $\mathcal{A}$, if he knows which node was the initial one and has good knowledge of the network's topology. The DirectionV probability distribution of Figure 4 shows a quite irregular structure, however with several unwanted peeks, which can be ascribed to the used *(pseudo) random number generator*. Otherwise, the DirectionV has the closest resemblance of the four basic choice functions to the desired uniform probability distribution. Unfortunately, the long term behavior, i.e. the distribution after a large number of e, is not as pleasant [1].

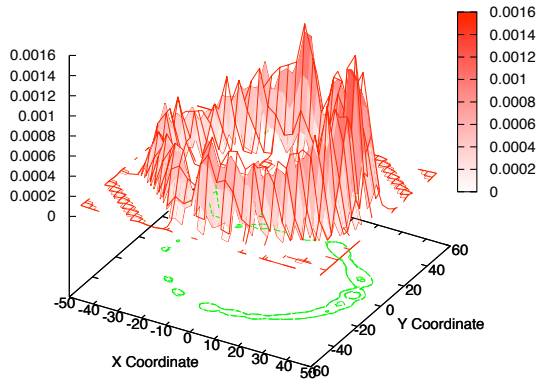

Figure 3. Evasive Data Storage with GNodeFurthest after 6 evasive steps.

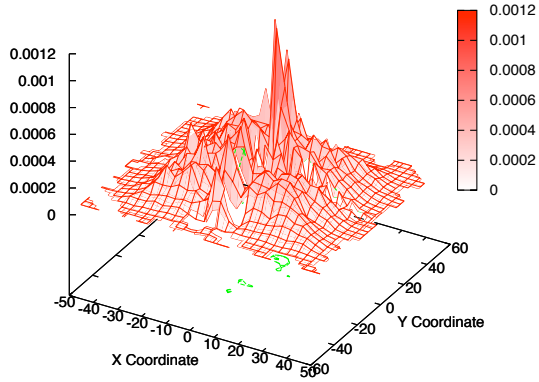

Figure 4. Evasive Data Storage with DirectionV after 6 evasive steps.

Overall, we have seen that none of the four basic choice functions does exhibit the wanted uniform distribution, and need to look for other functions that do

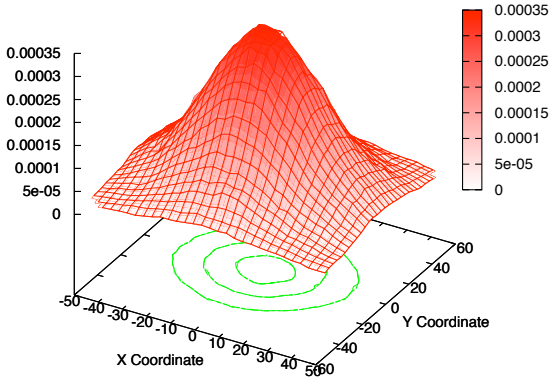

Figure 5. Evasive Data Storage with UVicinity after 18 evasive steps.

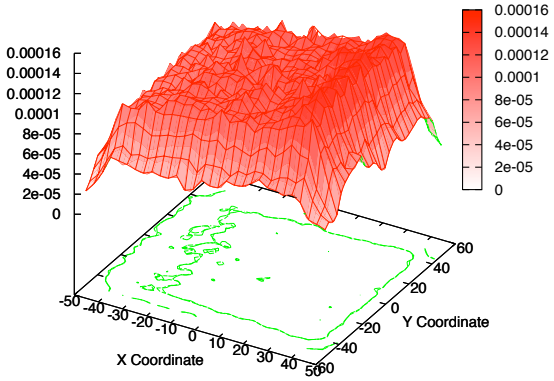

Figure 6. Evasive Data Storage with (DirectionV,$\frac{e}{2}$,UFurthest$k$) for $e = 18$ and $k = 2$.

offer such a property. Thus, we look at choice functions obtained from the combination mechanism discussed in the respective section above. To do so, we simulate the UVicinity function for 18 evasion steps, whose distribution hardly differs from the one obtained for 6 steps (compare 1 and 5), and the (DirectionV, $\frac{e}{2}$, UFurthest$k$) for the same number of steps. The respective result can be seen in Figure 6, which certainly shows that the probability distribution is the anticipated uniform one. Comparing this distribution with the UVicinity's one, it is clear that we can force an adversary to consider all instead of few nodes, when we employ the right choice function given the approximate size of the network. Thus we have proof indicating that choosing the right combined choice function can increase the success of data protection in the network, making the right choice function the core parameter for successful protection of data assuming Evasive Data Storage.

3.4 Extensions

Taking a look at the simple algorithm presented above for the Evasive Data Storage, it becomes clear that during data query the localization of data wandering the network can pose a problem. The simplest solution that we can provide is to flood the network, hence surely reaching every hot node storing data at that instance. A more elaborate scheme is to keep the base stations informed about the location of data wandering, thus avoiding the need for flooding. Either way, as long as the whole network is allowed to be used for wandering of data, the query cost will be high. Hence, we propose an easy solution to the problem, which bounds the region where data can wander to so called hot groups. Those groups consist of nodes that commit to store data evasively and only displace data to nodes belonging to the same group. With such a scheme, we reduce the query costs, as the number of nodes that need to be contacted is lowered, but still retain the advantages of Evasive Data Storage as long as the hot groups are not made ridiculously small. This notion is referred to as Location Bound Evasive Data Storage.

As another possible extension concerns the time data remains at a node. Introducing the Time Constraint Evasive Data Storage algorithm, we do allow for more control over actual *time* data resides at a hot node, before being considered for evasion. In the simple algorithm this is supposed to be some random time interval, but with the Time Constraint Evasive Data Storage algorithm a bound on the maximal time is imposed, thus making it possible to let data only remain a specified amount of time at a hot node.

All of the extensions to the simple algorithm for Evasive Data Storage are presented and discussed in more detail in the full version of this work [1].

At this point we want to address an important issue that is of significance to the whole idea of evasive storage regarding its security properties. Looking at an *active* adversary $\mathcal{A}$ that can takeover nodes in the network, it should be clear that he can strategically infect certain nodes with malicious code making them wait for data to be evaded into them. This simple setup renders the effectiveness of Evasive Data Storage to address data security very questionable. We investigated this issue by using simulations and confirmed that the larger e becomes, the higher the success probability of $\mathcal{A}$ collecting data through his setup malicious nodes. However, we have also showed in those simulations that a simple enhancement, namely *Data Splitting*, solves this deficiency of Evasive Data Storage in case of an active $\mathcal{A}$. The basic notion behind Data Splitting is to split data items into *splints* making it more difficult for $\mathcal{A}$ to successfully obtain data. This, of course, works at the expense of additional communication costs. The effectiveness is shown in scope of [1], and and it can be argued that keeping the frequency

of evasions low, the impact on the energy consumption can be kept at an acceptable level.

4 Conclusion

In this paper we introduced the notion of Evasive Data Storage and argued that it is an interesting method to combat adversaries that are able to perform node capture. Assuming an adversary has once identified a hot node, he cannot easily access that node's data at a later point in time with very high probability. This is in contrast to conventional storage mechanisms used today.

Using security enhancing techniques in sensor networks will most certainly in all cases be achieved through increased energy consumption, Evasive Data Storage is no exception to that. We can argue that there is a tradeoff between security and energy that in case of Evasive Data Storage demands for a very diligent choice of the available parameters. This adjustment is especially important when extension to the basic algorithm are considered too.

References

[1] P. M. Cholewinski. Evasive data storage in sensor networks. Diploma thesis, RWTH Aachen University, Laboratory for Dependable Distributed Systems, Germany, 2005.

[2] J. Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *Proceedings of the IEEE Conference on Security and Privacy in Communication Networks (SecureComm)*, 2005.

[3] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM Press, 2002.

[4] F. Howell and R. McNab. simjava: a discrete event simulation package for Java with applications in computer systems modelling. In *Proceedings of the First International Conference on Web-based Modelling and Simulation*, San Diego CA, Jan. 1998. Society for Computer Simulation.

[5] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61. ACM Press, 2003.

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

[7] 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 *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, Georgia, Sept. 2002.

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

[9] S. Zhu, S. Setia, and S. Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 62–72. ACM Press, 2003.