Catch Me (If You Can): Data Survival in Unattended Sensor Networks

Roberto Di Pietro[1] Luigi V. Mancini[2] Claudio Soriente[3] Angelo Spognardi[2] Gene Tsudik[3]

[1] Dipartimento di Matematica
Università di Roma Tre
dipietro@mat.uniroma3.it

[2] Dipartimento di Informatica
Università di Roma "La Sapienza"
{lv.mancini,spognardi}@di.uniroma1.it

[3] Computer Science Department
University of California, Irvine
{csorient,gts}@ics.uci.edu

Abstract

Unattended sensor networks operating in hostile environments might collect data that represents a high-value target for the adversary. The unattended sensor's inability to off-load – in real time – sensitive data to a safe external entity makes it easy for the adversary to mount a focused attack aimed at eliminating certain target data. In order to facilitate survival of this data, sensors can collectively attempt to confuse the adversary by changing its location and content, i.e., by periodically moving the data around the network and encrypting it.

In this paper, we focus on data survival in unattended sensor networks faced with an adversary intent on surgically destroying data which it considers to be of high value. After motivating the problem and considering several attack flavors, we propose several simple techniques and provide their detailed evaluation.

1. Prelude

Sensors and sensor networks have generated a veritable flurry of research activity in the past decade. Much of the prior research concentrated on so-called *Wireless Sensor Networks* (WSNs) and included: routing, security, power-awareness, data abstraction as well as many other aspects of WSN operation.

One common assumption in most prior WSN security research has been that data collection is performed in a more-or-less real time fashion. In other words, a trusted entity (such as a collector or a sink) is assumed to be present. A variation of this assumption is that the this entity is able to *query* the WSN in real time and obtain a reply. While it may well be true that many – or even most – sensor networks operate in such general settings, there remains a segment of WSNs and their applications that do not fit within the real time data collection model. Such networks are sometimes referred to as *Unattended WSNs* or UWSNs, for short. There are also hybrid UWSNs where the set of nodes, in addition to sensors, includes some small number of collector nodes, each serving as a temporary repository for their constituent sensors. The real sink later obtains data directly from the collector nodes.

In this paper, we are interested in *homogeneous* UWSNs, meaning that the network is composed of peer sensors. Furthermore, we focus primarily on UWSNs operating in hostile, or at least untrusted, environments where the adversary has relatively free reign. In particular, the adversary is assumed to be single-mindedly focused on destroying certain target data collected by sensors. We elaborate on this below.

There are a number of real and easily imagined hostile environments that match the above description. Consider, for example, a network of nuclear emission sensors deployed in a recalcitrant country (say, under an international treaty) in order to monitor any potential nuclear activity. Another example is an underground sensor network aimed at monitoring sound and vibration produced by troop movements (or border crossings). One can also imagine an airborne sensor network tracking fluctuations in air turbulence and pressure in order to detect enemy aircrafts. Among the features that unify these examples is the likely presence of a powerful – yet careful – adversary. Informally speaking, we say that the adversary is *powerful* if it can subvert a number of sensors at will, while it is considered *careful* if it wishes to remain undetected in the process.

In such settings, one of the biggest challenges is to ensure data survival for long enough that it can be collected by the itinerant sink. Clearly, if the adversary is unable to break into (i.e., compromise) a single sensor or inhibit communication between a sensor and an eventual collector or sink, it has no hope of destroying the data. However, we envisage a more realistic adversary that is aware of the origin(s) of target data and is also assumed capable of compromising any sensor it chooses, up to a specific threshold (fraction or absolute number) of sensors, within a certain

time interval. Such an adversary has been studied in the cryptographic literature where it is usually referred to as a *Mobile Adversary* [13]. An entire branch of cryptography, called *Proactive Cryptography* has been dedicated to developing cryptographic techniques (e.g., decryption and digital signatures [7, 8]) that remain secure in the presence of a mobile adversary. Although our adversary models are identical, the UWSN application domain is very different from that in proactive cryptography (as described below), thus motivating radically different solutions.

UWSNs have also recently been considered in the context of minimizing storage and bandwidth overhead due to data authentication in the presence of a powerful adversary [12]. The resulting forward-secure aggregate authentication techniques can efficiently provide for so-called forward security, i.e., having compromised a sensor, the adversary is unable to modify any data collected by the sensor prior to compromise (other than by deleting all accumulated data). Our focus is quite different: we assume that the adversary is actively *hunting* certain data and is not afraid to delete/erase any data it finds.

1.1 Contributions

The work described in this paper makes three main contributions:

1. Problem Exposure: we believe that this paper is the first to identify the problem of data survival in UWSNs in the presence of a powerful mobile adversary. Besides exposing the problem, we probe a number of relevant issues having to do with the exact power of the adversary and the resilience of the network.

2. Simple Technique & Analysis: we propose a straightforward non-cryptographic technique aimed at hiding the data from the adversary. Our evaluations show a surprising degree of data survival even when the adversary is relatively capable and the time between successive sink visits is relatively long. However, we also demonstrate the eventual futility of hiding data location without hiding its contents.

3. Raising the Stakes: after observing that the simple technique has certain basic limitations, we gradually introduce a more advanced approach based on standard cryptographic tools.

Organization of this paper: Section 2 overviews related work, followed by Section 3 which introduces our environment assumptions. Then, Section 4 explores potential strategies for both the UWSN and the adversary and Section 5 provides both analyses and simulation results for various strategies. Section 6 considers using replication as an additional survival aid and demonstrates a set of corresponding simulation results. Section 7 sketches how the introduction of cryptography helps data survival. The paper ends in Section 8 with discussion and directions for future work.

2. Related Work

Since WSNs share many features with Mobile Ad Hoc Networks (MANETs), it is not surprising that some related work comes from the MANET research literature. The problem of data availability in MANETs has been studied extensively in the relatively benign context of communication faults and network partitions. This research thread aims to maintain data availability to any MANET node, even if the network is fragmented. For example, Hara, et al. [10] introduced some simple and effective algorithms to replicate data in MANETs, such that a node in a disconnected partition can access any data with high probability. The system also provides some means to deal with replica consistency (in case of updates to the original data) and a simple location management technique to guarantee that nodes access the closest replica.

In another related result, Giannuzzi, et al. [9] studied data availability through replication in paritioned MANETs. This work shows that the probability of accessing certain data is dependent not only on the number of replicas but also on the network density as well as on the nodes' transmission radius.

Chessa, et al. [5] introduced a distributed data storage approach for MANETs, based on the peer-to-peer paradigm. This approach provides support for creating and sharing files under a write-once model; it also ensures data confidentiality and dependability by encoding files in a Redundant Residue Number System (RRNS).

One more recent result addressed data availability in WSNs [11]. It develops a way to maximize the amount of data recovered at the sink and shows how the proposed scheme improves data availability when a portion of the network is unavailable due to natural disasters.

Finally, another set of results [4, 14] leverages the availability of multiple paths between end-nodes to statistically improve data confidentiality and availability in hostile MANET settings, in the presence of both insider and outsider adversaries.

3. System Assumptions

In this section we describe our assumptions about the network and the adversary.

3.1 Network Assumptions

The envisaged UWSN is assumed to operate as follows:

- Sensors are programmed to collect data periodically. There is a fixed global periodicity parameter p which denotes the time interval between successive sensing actions. Note that this rules out certain sensor environments which operate on query basis, i.e., where sensors only obtain data when explicitly requested to do so.
- Each sensor collects a single unit of data for each interval. In an UWSN composed of n sensors, we say, each sensor s_i collects data d_r^i for interval r.
- The network is unattended. There exists a parameter $q > p$ which denotes the maximum time between successive visits of the sink or collector. (We use the term sink from here on to mean both). We use $v = q/p$ to represent the maximum number of sensing intervals between successive sink visits.
- The network is connected at all times. Any two sensors can communicate either directly or indirectly, via other sensors. Although we use the term UWSN, we make no assumption about the wireless medium. (The network can, in fact, be connected with wires.)
- There is no power constraint. At least initially, we are not concerned with power consumption of various defensive techniques. (This assumption will be re-considered later.)
- Ample storage. Each sensor is equipped with enough storage to accommodate $O(v)$ sensed data.

As seen from our assumptions, even apart from its unattended nature, we are considering a kind of a sensor network not typically encountered in the research literature.

3.2 Adversary Model

We now focus on the description of the anticipated adversary. We refer to it as $\mathcal{A}$ from here on.

- Compromise power: $\mathcal{A}$ is capable of compromising at most k out of the total of n sensors during one collection interval. The parameter k may be absolute, i.e., an integer, or relative, i.e., a fraction of n – the total number of sensor nodes. When $\mathcal{A}$ compromises a node, and for as long as it remains in control of that node, we assume that it reads all of memory and monitors all incoming and outgoing communication.
- Network knowledge: $\mathcal{A}$ knows the composition and the topology of the network; even if the network topology changes over time.
- One chance to kill: in the time between two successive sink visits, $\mathcal{A}$ can erase only one data unit from only one sensor it compromises. (Erasing any more than that raises an alarm on the sink; we will re-consider this assumption later on.)
- No interference: except for the above, $\mathcal{A}$ does not interfere with communications of any sensor and does not *modify* any data collected by, or stored on, any sensors it compromises.
- Stealthy operation: $\mathcal{A}$'s movements between intervals are unpredictable and untraceable. Specifically, it is impossible to detect when and if the adversary ever *visited* (compromised) a particular sensor.
- Atomic movement: $\mathcal{A}$ moves in one fell swoop, i.e., at the end of each interval, it selects at most k nodes for the next round and migrates to them in one monolithic step. Note that the two sets of compromised nodes may intersect or even be the same.
- Leave nothing behind: as it moves from one set of k nodes to the next, $\mathcal{A}$ leaves no trace behind.

Table 1 summarizes the notation used in the remainder of the paper.

n	size of the UWSN
$\mathcal{N}$	set of the sensor nodes of the UWSN
i, j	sensor indexes
s_i	sensor i
r	round index
d_r^i	sensed data collected by sensor i at interval r
x	target data
$\bar{s}_r$	sensor that at round r stores x
v	number of rounds between successive sink visits
C_r	set of compromised sensors (controlled by $\mathcal{A}$) at round r
k	size of C_r; we assume it to be constant

Table 1. Notation Summary

4. Strategies and Design Space

As discussed earlier, our main goal is to ensure survival of data collected by some sensor s_i at interval r. (Recall that survival means that this data is eventually delivered to the sink.) $\mathcal{A}$ picks both s_i and r. Unfortunately for us, since $\mathcal{A}$ does not reveal the data it is interested in *erasing*, we know neither of these values, other than $1 \leq i \leq n$ and $0 \leq r \leq v$.

As a consequence, we have to assume the worst, meaning that $\mathcal{A}$ will target some data collected during round 0,

immediately following the sink visit. (Hence, $\mathcal{A}$ has maximum time to look for the target data.) Similarly, we must assume that all data is potentially targeted by $\mathcal{A}$, i.e., every d_r^i is equally likely to be the adversary's target.

We now consider several survival strategies. From the network's perspective, there are three intuitive options:

- DO-NOTHING. The easiest thing is to do nothing: simply leave data resident on the sensor that collected it and wait for the sink. In this case, $\mathcal{A}$ succeeds quickly by compromising the target sensor and erasing the target data.
- MOVE-ONCE. A trivial alternative is for each sensor – right after collection – to move the newly obtained data to some other randomly picked sensor. The data then remains at its new "home" until the next sink visit.
- KEEP-MOVING. A more laborious option is to move data continiously, i.e., at every interval, each sensor moves each data item individually to another randomly chosen sensor.

No matter what survival strategy is used, we must assume that $\mathcal{A}$ is fully aware of it. Knowing the network's survival strategy, $\mathcal{A}$ can formulate an attack strategy that maximizes its chances of destroying the target data.[1] We consider the following possibilities:

- LAZY: The easiest thing for $\mathcal{A}$ is to do nothing: compromise k nodes at the start and stay put. In other words, $C_{r+1} = C_r$ for $0 < r \leq v$. This attack strategy is not sensible if the survival strategy is, for example, DO-NOTHING or MOVE-ONCE. (This is because $\mathcal{A}$ might never capture the target data, no matter how large v is). It is more effective in other cases, as will be seen below.
- FRANTIC: Another extreme is for $\mathcal{A}$ to move at the beginning of each interval $r+1$ to a set C_{r+1} of k nodes randomly chosen among the set $\mathcal{N} \setminus C_r$. In other words, $C_{r+1} \cap C_r = \emptyset$ for $0 < r \leq v$.
- SMART: An alternative counter-strategy is for $\mathcal{A}$ to pre-select two sets of nodes: $\mathcal{C}^0$ and $\mathcal{C}^1$, each of size k. Then, $\mathcal{A}$ simply alternates control between these two sets at each interval. In other words, $C_r = \mathcal{C}^{r \bmod 2}$ for $0 < r \leq v$. This attack strategy is clearly a poor match for DO-NOTHING and MOVE-ONCE (but works well for KEEP-MOVING, as discussed below.)

It is easy to see that not all attack-survival strategy combinations make sense. Table 2 illustrates the possibilities.

[1] The reverse is not true, i.e., we *do not* assume that the adversarial strategy is known to the UWSN.

Table 2. Viability of Survival and Attack Strategies

	Attack Strategy:		
Survival Strategy:	LAZY	FRANTIC	SMART
DO NOTHING	*NO*	YES	*NO*
MOVE ONCE	*NO*	YES	*NO*
KEEP MOVING	YES	YES	YES

A "YES" label in a table cell implies that the corresponding combination of defense and attack strategies is viable. Whereas, a "NO" means that the corresponding combination is not sensible, at least for $\mathcal{A}$. For example, consider a DO-NOTHING defense: if the adversary is LAZY, it stands little chance of finding target data (unless it gets lucky and C_1 includes the sensor that collected target data).

In the rest of this paper, we will investigate viable combinations reflected in Table 2.

Encryption. An important issue is whether UWSN nodes can use encryption. If encryption is available, nodes can hide not only the collected data itself but also the identity of the sensor that collected it, as well as the round identifier. (Of course, this does not help at all with the DO-NOTHING defense strategy.) Use of encryption is a natural choice, however it comes with certain non-negligible costs, including key management and encryption overhead. Encryption also motivates certain assumptions and technicalities which we discuss later in the paper (see Section 7). In summary, we do not mandate the use of encryption but treat it as an option. Our initial investigation focuses on encryption-free UWSNs.

Replication. Another design dimension is replication, i.e., whether sensors create multiple copies of sensed data before moving it to other locations. Replication has some obvious advantages and drawbacks. The main advantage is increased chances of data survival, while the main drawback is increased storage and communication overhead. If replication is used, the next issue is its degree, i.e., how many replicas are sufficient to guarantee a certain probability of survival, given a particular attack strategy. We begin examining the case of no replication and then proceed to consider replication as an additional survival aid.

5. Evaluation

We now investigate in more detail some possible survival and attacks strategies.

5.1 DO-NOTHING and MOVE-ONCE

For the sake of completeness, we start with the DO-NOTHING defense strategy. As the name suggests, there is not much to evaluate. Clearly, $\mathcal{A}$ can attain its goal very quickly and certainly: it learns the identity of $\bar{s}_0$ at the end of round 0. Then, at round 1, it compromises any set of nodes that includes $\bar{s}_0$, and proceeds to delete x.

We now turn to the MOVE-ONCE strategy, as described in Section 4. Even though $\mathcal{A}$ knows $\bar{s}_0$ identity at the end of round 0, it has no knowledge of $\bar{s}_1$ – the node selected by $\bar{s}_0$ as *home* for data x. Since any sensor is equally likely to be $\bar{s}_1$, $\mathcal{A}$ cannot do better than selecting the next compromised set C_1 at random.

The best attack strategy for $\mathcal{A}$ is FRANTIC. The reason for this is intuitive: If $\mathcal{A}$ chooses to be LAZY, its probability of success is fixed at $\frac{k}{n}$, since it only has one round to capture x. Whereas, if $\mathcal{A}$ adopts the SMART strategy, since data does not continuously move around, the probability of success, though higher, is also fixed at: $\frac{2k}{n}$. (Recall that $\mathcal{A}$ alternates between compromising two k-sized fixed sets.) Assuming a FRANTIC adversary, if $\bar{s}_1 \in C_1$, then $\mathcal{A}$ wins at round 1. Otherwise, it proceeds, round-by-round, to select a new set C_r such that $C_r \cap (C_1 \cap ... \cap C_{r-1}) = \emptyset$. Trivially, it takes $\lceil \frac{n}{k} \rceil$ rounds for $\mathcal{A}$ to inspect the memory of all UWSN nodes. Thus, after $\frac{n}{2k}$ expected rounds, $\mathcal{A}$ succeeds in finding and deleting x. Moreover, if $\frac{n}{k} < v$, $\mathcal{A}$ is guaranteed to win the game, i.e., delete x before the next sink visit.

Assuming, for simplicity, that $\frac{n}{k} = c$ (for some integer c), the probability of the event: G_r ="$\mathcal{A}$ finds x at round r" ($1 < r \leq c$), conditioned on the event F_{r-1}="$\mathcal{A}$ does not find x in prior $r-1$ rounds", is:

$$Pr[G_r|F_{r-1}] = \frac{k}{n-(r-1)k} \quad (1)$$

Also, the probability $Pr[G_r]$ can be derived as:

$$Pr[G_r] = \frac{k}{n} \quad (2)$$

It might seem that $\mathcal{A}$'s chances to win the game are low in the first rounds, while increasing as rounds go by. However, in every round, $\mathcal{A}$ has the same probability (k/n) of success. For justification of Equation 2, please refer to [6].

5.2 KEEP-MOVING

In this section, we investigate the effectiveness of the KEEP-MOVING defense strategy – at the end of each interval, every sensor moves its data to a randomly selected sensor. Algorithm 1 shows the corresponding pseudo-code executed by the adversary who is following one of the three strategies outlined in Section 4.[2] The only difference between them is the particulars of step 1, i.e., how $\mathcal{A}$ selects C_r. Recall that: in LAZY strategy C_r stays the same, in FRANTIC – it keeps changing each round, and, in SMART – $\mathcal{A}$ alternates between two sets of nodes, each of size k.

Algorithm 1: KEEP-MOVING

```
   /* start round 0 */
   all nodes sense their values
   each node exchanges data with others
0  A learns s̄_0 and x
   /* end round 0 */
   SET z=min(v, n/k)
   SET found=FALSE
   for ((r ← 1 to z) and (not found)) do
       /* start round r */
1      select C_r  /* new set of nodes to compromise */
2      compromise C_r and release C_{r-1}
3      if (x found on some s_i ∈ C_r) then
3.1        delete x
3.2        SET found=TRUE
       all nodes sense their values
       each node exchanges data with others
4      if (x received by some s_i ∈ C_r) then
4.1        delete x
4.2        SET found=TRUE
       /* end round r */
```

We now analyze the three aforementioned attack strategies. Referring to Algorithm 1, for each round $r > 0$, let $\bar{s}_r$ denote the sensor storing x until step 4 and let $\bar{s}_{r+1}$ be the sensor that receives and stores x in step 4 (the same sensor will keep x until step 4 of the following round).

5.2.1 LAZY

We start with the LAZY attack strategy: at round 1, $\mathcal{A}$ randomly chooses a set C_1 and keeps it the same for all subsequent rounds. This strategy is not completely stupid since $\mathcal{A}$ exploits the fact that data (including the target x) is constantly moving among sensors. $\mathcal{A}$ wins the game if some node currently storing x chooses one of the nodes in C_1 as the new recipient of x.

Referring again to Algorithm 1, it seems that $\mathcal{A}$ always has two chances to delete x in a given round: either (1) x is stored in one of the sensors it currently controls, or (2) x gets fortuitously moved by some non-compromised sensor to another sensor which happens to be in C_r. However, with a LAZY approach, the first chance only occurs once, at round 1. Thereafter, since $\mathcal{A}$ does not move ($C_r = C_1,\ r >$

[2]Note that the algorithm depicts the worst-case scenario, i.e., we assume that the target data x is collected in the interval immediately following the sink visit. Consequently, $\mathcal{A}$ has v rounds at his disposal before the next sink visit.

1), it can only take advantage of the second chance, i.e., some sensor moving x to a node in C_1.

We can thus quantify the probability that $\mathcal{A}$ wins during the first round:

$$Pr[\textit{success}] = \frac{k}{n} + \left(1 - \frac{k}{n}\right)\frac{k}{n} \tag{3}$$

Intuitively, $\mathcal{A}$ wins in $r = 1$, if $\bar{s}_1 \in C_1$ or if $\bar{s}_2 \in C_1$. That is, $\mathcal{A}$ wins if C_1 contains the node that received x in round 0 from $\bar{s}_0$ ($\bar{s}_1 \in C_1$), or if a node in C_1 receives x from $\bar{s}_1$ in round 1 ($\bar{s}_2 \in C_1$). $Pr[\{\bar{s}_1 \in C_1\}] = \frac{k}{n}$, $Pr[\{\bar{s}_2 \in C_1\}]$ is the same, conditioned on $\bar{s}_1 \notin C_1$, that is $Pr[\{\bar{s}_2 \in C_1\}] = Pr[\{\bar{s}_2 \in C_1 | \bar{s}_1 \notin C_1\}] \cdot Pr[\{\bar{s}_1 \notin C_1\}]$.

Let P_1 be the probability that $\mathcal{A}$ does not win in round one: we have that P_1 is the complement of equation 3, that is

$$P_1 = 1 - Pr[\textit{success}] = \left(1 - \frac{k}{n}\right)^2 \tag{4}$$

Now, the probability that the LAZY $\mathcal{A}$ fails for v consecutive rounds, i.e., the probability that x survives v rounds is:

$$P_L(v) = P_1 \cdot P_2^{v-1} \tag{5}$$

$$\text{where } P_1 = \left(1 - \frac{k}{n}\right)^2 \text{ and } P_2 = \left(1 - \frac{k}{n}\right)$$

For a justification of equation 5, please refer to [6].

As a result, the probability that the LAZY $\mathcal{A}$ wins at round $v > 1$ is

$$\overline{P_L}(v) = P_1 \cdot {P_2}^{v-2} \cdot \overline{P_2} \tag{6}$$

where $\overline{P_2} = \frac{k}{n}$. By sampling equation 5, we can get an idea of the effectiveness of this strategy. For example, in a UWSN of 100 sensors and $k = 10$, the probability that x survives at least 5 rounds, $P_L(5) = 0.5$, $P_L(10) = 0.31$, and $P_L(20) = 0.11$. A more detailed qualitative evaluation of $P_L(v)$ is reported upon in Section 5.3.

5.2.2 FRANTIC

In this section, to evaluate the effectiveness of the frantic strategy, we introduce the following claim.

The probability that a FRANTIC $\mathcal{A}$ fails in v consecutive rounds is:

$$P_f(v) = P_1 \cdot P_2^{v-1} \cdot P_3^{v-1} \tag{7}$$

where

$$P_1 = \left(1 - \frac{k}{n}\right)^2, P_2 = \left(1 - \frac{k}{n}\right) \text{ and } P_3 = \left(1 - \frac{k}{n-k}\right)$$

The justification for equation 7, can be found in [6].

As a result, we can state the probability that the FRANTIC $\mathcal{A}$ wins at round $v > 1$ is:

$$\overline{P_f}(v) = P_1 \cdot (P_2 P_3)^{v-2} \cdot (\overline{P_2} + P_2\overline{P_3}) \tag{8}$$

where $\overline{P_3} = \frac{k}{n-k}$ and $\overline{P_2} = \frac{k}{n}$. To get an idea of the effectiveness of the FRANTIC attack strategy, consider that in a UWSN with $n = 100$ sensors and with $k = 10$, the probabilities that x survives for 5, 10 and 20 rounds are, respectively: 0.33, 0.11 and 0.012. Whereas, the corresponding probabilities for the LAZY $\mathcal{A}$ are, respectively: $0.5, 0.31$ and 0.11.

5.2.3 SMART

As illustrated above, the FRANTIC attack strategy is quite effective. However, it requires $\mathcal{A}$ to constantly move around which involves some amount of effort and risk of detection.

To minimize efforts – and knowing that the UWSN is employing the KEEP-MOVING defense – our adversary might adopt the SMART strategy, as outlined in Section 4. Intuitively, because of the constant data mobility inherent to the KEEP-MOVING defense, a SMART $\mathcal{A}$ achieves the same probability of success as a FRANTIC $\mathcal{A}$. While this claim might seem counter-intuitive, consider the following observation:

> Regardless of how (randomly or otherwise) a SMART $\mathcal{A}$ selects the two sets of nodes $\mathcal{C}^0$ and $\mathcal{C}^1$, the choice of $\bar{s}_{r+1}$ (the new "host" of the target data) is made uniformly, at random.

Therefore, the only condition for the result in Section 5.2.2 to apply to the SMART $\mathcal{A}$ is that $C_r \bigcap C_{r+1} = \emptyset$.

5.3 Simulation Results and Discussion

To confirm the above analytical results, we developed a UWSN simulator and obtained some simulation results. In this section, we present these results for the three attack strategies and discuss some cost-related issues.

Our simulator faithfully follows Algorithm 1 in Section 5.2. For every simulation run, we set the size of the network n, the size of set C_r, and the attack strategy. We record the round at which $\mathcal{A}$ captures the target value x. This allowed us to obtain an estimate of the effectiveness of the three attack strategies.

Fig. 1 shows the probability of x surviving r rounds with k (number of nodes $\mathcal{A}$ can compromise in a single round) varying between $2, 5$ and 10. In all of the following simulations, the network size is fixed to 100 nodes. We stress the importance of parameter k. Increasing k provides an advantage to the adversary that is more than linear when k is small. For instance, between rounds 5 and 10, the survival probability when $\mathcal{A}$ controls $k = 5$ nodes is three times the corresponding probability for $k = 10$. For number of rounds between 10 and 25, the probability is at least six times bigger, even if asymptotically both probabilities converge to zero, i.e., $\mathcal{A}$ eventually wins as r grows. Note that

scheme	*#msg at round r*	*#msg up to round r*	*#msg in memory at round r*	*#msg received at round r*
Move-Once	n	$r \cdot n$	$Pr[\exists\ s_i \text{ s.t. } L_i^r \geq r + \sqrt{nr}] \leq e^{-r/2+\ln n}$	$O(\ln n)$
Keep-Moving	$r \cdot n$	$\frac{r(r-1)}{2}n$	$Pr[\exists\ s_i \text{ s.t. } L_i^r \geq 2er] \leq 2^{-r+\ln n}$	$Pr[\exists\ s_i \text{ s.t. } M_i^r \geq 2er] \leq 2^{-r+\ln n}$

Table 3. Overhead comparison

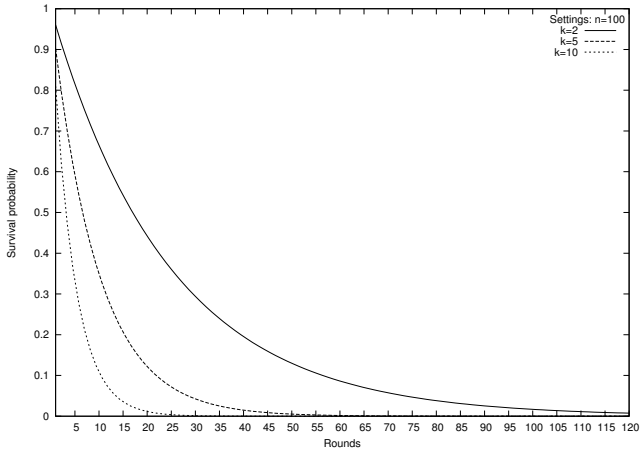

Figure 1. Probability of x survival for $n = 100$ and $k = 2, 5, 10$, respectively.

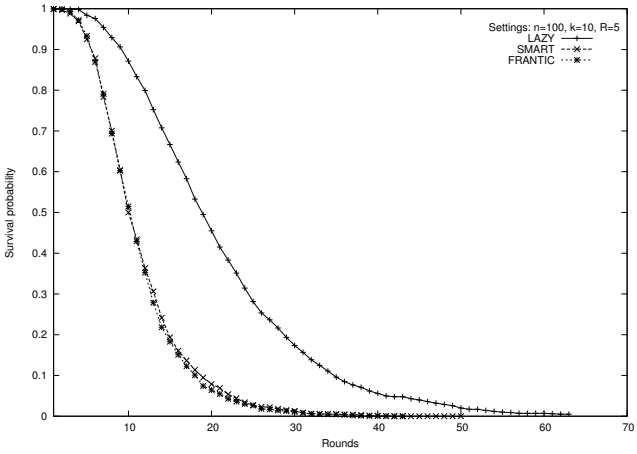

Figure 2. Comparison between the three types of adversary: $n = 100, k = 10, R = 5$

$k = 10$ implies dealing with $\mathcal{A}$ controlling 10% of the network, quite a powerful $\mathcal{A}$.

Table 3 shows the overhead for the defense strategies. The first row corresponds to MOVE-ONCE, and the second – to KEEP-MOVING. The table illustrates that the overhead in terms of message transmissions is quite high for KEEP-MOVING. Regarding the overall storage load and the number of messages received in a single round, the table shows the probability bound which, in turn, demonstrates that these solutions viable.[3]

6. Replicas

Since our main goal is to keep x alive for v rounds, introducing replicas in the network will increase the probability that the sink will find at least one surviving copy of x. If s_i, after sensing d_r^i, creates and distributes R copies, $\mathcal{A}$ must delete all copies to win the game. As shown below, the probability of success of $\mathcal{A}$ would decrease as R increases. However, the drawback is the commensurate R-fold increase in the number of messages in the network, which affects memory and energy requirements.

Let $X_{i,j} = 1$ denote the fact that the i-th replica survives up to j^{th} round, and $X_{i,j} = 0$ denote the fact that $\mathcal{A}$ erased it. According to equation 7, we have:

$$Pr[X_{1,j} = 1] = P_1 \cdot P_2^{j-1} \cdot P_3^{j-1}$$

Now, we need to evaluate the probability $\overline{P_R^v}$ of having no replicas surviving up to round v. We have:

$$\overline{P_R^v} = Pr[X_{1,v} = 0 \wedge \ldots \wedge X_{R,v} = 0] = Pr[X_{1,v} = 0]^R = \\ = (1 - Pr[X_{1,v} = 1])^R = (1 - P_1 \cdot P_2^{v-1} \cdot P_3^{v-1})^R$$

The probability of at least one (of R) replicas surviving to round v is:

$$P_R^v = 1 - \overline{P_R^v} = 1 - (1 - P_1 \cdot P_2^{v-1} \cdot P_3^{v-1})^R \quad (9)$$

6.1 Further Simulations

As shown by Equation 9 and the results of our simulations, replicas act as an effective additional counter-measure to the focused erasure pursued by $\mathcal{A}$. However, replication is only viable when the network is capable of handling an R-fold increase in storage and transmission costs.

Figure 2 shows the probability of $\mathcal{A}$ succeeding in a UWSN where $n = 100$, $k = 10$, and $R = 5$. As in the case of no replication, the LAZY strategy is somewhat less efficient than either SMART or FRANTIC strategies.

[3] For the proof of bounds in Table 3, refer to [6].

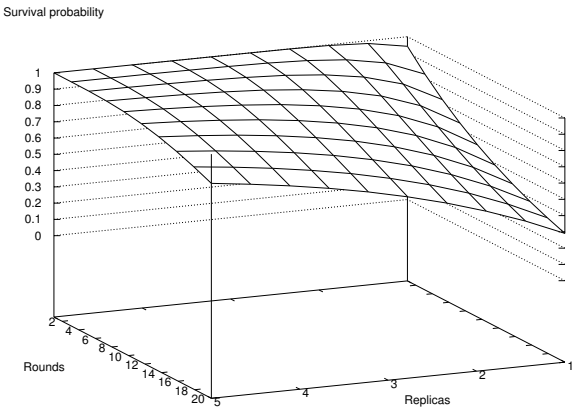

Figure 3. Survival probability, according to the degree of replication R, with n=100, k=3

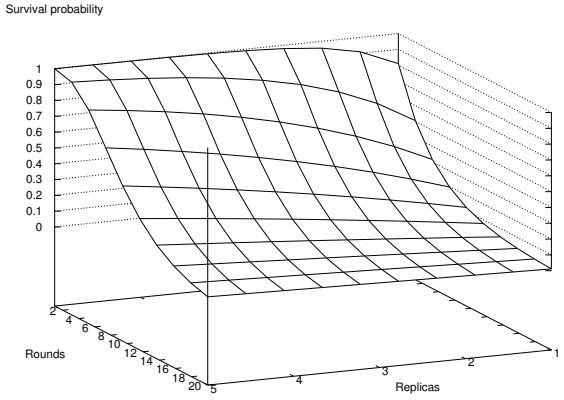

Figure 4. Survival probability, according to the degree of replication R, with n=100, k=10

Comparing the FRANTIC curve in Figure 2 with the third curve in Figure 1 (for $k = 10$), the benefit of replicas can be easily appreciated. For example, probability of x surviving 5 rounds when data is not replicated, is 0.33, while it is 0.87 for $R = 5$. As expected, the differences between the two strategies diminish as the number of rounds increases.

Figure 3 demonstrates how the probability of data survival changes with respect to the degree of replication in an UWSN with $n = 100$ and $k = 3$. In particular, after 20 rounds, survival probability increases from 0.29 when no replication is used to 0.82 when $R = 5$.

In Figure 4, we consider a more capable $\mathcal{A}$ that is able to compromise 10% of the network nodes at each round. With such an adversary, the survival probability after 20 rounds becomes almost zero regardless of the degree of replication. However, it is worth noting that data replication is still effective during the first few rounds: for example, the probability of x surviving after 6 rounds, ranges from 0.26 when x is not replicated, to almost 0.79 when $R = 5$. Hence, if the number of rounds exceeds a certain threshold, replication does not substantially help in data survival. However, if the number of round between successive sink visit is reasonably small, replication does offer substantially higher probability of survival even against a more capable adversary.

The main conclusions that can be drawn from our results is the surprising degree of data survival even when the adversary is relatively capable with respect to the time between successive sink visits. Furthermore, for small values of v, benefits of replication are magnified as k increases.

7. Encryption

As mentioned earlier in the paper, the use of encryption is an obvious aid in data survival. However, it involves certain costs. We now look into several issues associated with the use of encryption and sketch out some possible strategies for integrating encryption with the defense (and attack) strategies discussed above.

Note that we are concerned with encryption as a means of obtaining data privacy, i.e., hiding the origin and the content of data. In this context, we are not concerned with data authenticity, since our adversary's goal is strictly to eliminate (and not to modify) data. However, since data modification can be viewed as a form of data erasure, we assume that, data authentication is provided before encryption is applied to data. More concretely, we assume that all data is encrypted using Plaintext-Aware Encryption [2] whereby any modification (without knowledge of the secret key) will produce gibberish upon attempted decryption.

Furthermore, we assume that, regardless of encryption particulars, encryption is always randomized, which (informally) means that given two encryptions under the same key, it is infeasible to determine whether the corresponding plaintext messages are the same.

As usual, the choice of public key or conventional techniques is the main variable when it comes to introducing encryption. We prefer to remain agnostic with respect to this

choice and briefly sketch out approaches for both cases.

7.1 Public Key Encryption

Public key encryption is typically avoided in the sensor network literature since its higher cost is a poor match for low-cost sensors. However, for the sake of completeness, we sketch out a possible setting:

- The sink has a public key, PK_{sink}, known to all sensors.
- As soon as a sensor s_i collects a unit of data d_r^i (at round r), it encrypts it to produce $E_r^i = E(PK_{sink}, d_r^i, r, etc.)$.
- The rest is identical to Algorithm 1.

Notice that sensors have no secret keys of their own – they merely use the sink's public key to encrypt data. However, $\mathcal{A}$ is greatly handicapped by this simple approach since it can no longer distinguish the target data from any other encrypted data it finds on compromised sensors.

Without access to the sinks's decryption key (SK_{sink}), the only way $\mathcal{A}$ can attempt to detect target data is by trying to encrypt its duplicate under the sink's public key, PK_{sink}. This is, however, where our randomized encryption assumption comes in handy. Randomized encryption basically assumes that each encryption operation involves generating a one-time random number and somehow "folding" it into the plaintext (e.g., as in OAEP+ [15]) such that, without knowledge of that unique random number, it is infeasible to re-create the same ciphertext. Consequently, $\mathcal{A}$ is completely unable to distinguish among (as far as which sensor encrypted them, what values are encrypted and/or at what interval) different encrypted values it finds on compromised sensors.

Another interesting feature of public key encryption is that some techniques (e.g., [1]) allow what is referred to as *re-encryption*. This means, in our setting, that a sensor which receives already-encrypted data from another sensor, can re-encrypt that data such that the previous sensor would be unable to recognize its own encrypted data thereafter. The re-encryption operation does not involve encrypting something twice – it is usually light-weight, only requiring some minor operations on already-encrypted data. The use of re-encryption might benefit us considering that $\mathcal{A}$ as it breaks into sensors at each round r, might attempt to copy and remember the encrypted values that sensors in C_r generate, encrypt and move out during the same round. (That way it could detect them later.) If all sensors re-encrypt all values they receive from others, $\mathcal{A}$ would be placed at a further disadvantage.

7.2 Conventional Encryption

The construction needs to be somewhat different with conventional encryption.

- Each sensor s_i shares a unique initial pairwise key k_0^i with the sink.
- As soon as s_i collects a unit of data d_r^i (at round r), it encrypts it to produce $E_r^i = E(k_r^i, d_r^i, r, etc.)$.
- Then, s_i computes $k_{r+1}^i = OWF(k_r^i)$ where $OWF()$ is a cryptographically suitable one-way function, such as SHA-2.
- The rest is identical to Algorithm 1.

Since conventional encryption is invertible, we need to worry about what happens when $\mathcal{A}$ compromises a given sensor s_i. Suppose that a particular s_i originally generated the target data x. If we do not change the key (under which x was encrypted), our adversary will win the game with almost the same probability as in the case with no encryption. To see this, consider that the adversary would simply collect and remember all (conventional) keys of all sensors it compromises. Then, when it breaks into a particular sensor, it can essentially decrypt most (or all) of the encrypted data it finds and eventually find the encrypted version of x.

Thus, we need a property commonly referred to as *Forward Security* [3]. This is the reason for *evolving* the key at each round, using the $OWF()$ function. A minor issue is how the sink would decrypt data: this does not pose a problem since the sink has all initial keys of the form k_0^i. It can then attempt to decrypt a given encrypted message using all $n * v$ possible keys. While this might seem excessive, we point out that conventional encryption is very inexpensive and the sink is assumed to have no computational constraints.

7.3 Effects of Encryption

Based on the above discussion, we claim that, regardless of the encryption type (public key or conventional), our adversary has equally diminished capacity to detect (and erase) target data as it inspects the memory of compromised sensors.

At the same time, we probably need to re-examine our adversarial model. Recall that $\mathcal{A}$ has *one-chance-to-kill* as described in Section 3.2. We need to re-evaluate this feature mainly because the adversary is now unable to determine what data to erase. Specifically, it has no idea what (encrypted) data originated at which sensor. Hence, *one-chance-to-kill* no longer applies.

One possible modification is to allow $\mathcal{A}$ to erase up to a certain number (say, b) of encrypted data values. This

immediately motivates a different analytical model, for each of the defense and attack strategy combinations explored in this paper. This clearly represents an exciting avenue of research which is indeed currently on-going. However, due to space limitations we do not report our preliminary results but defer them to the later (full) version of this paper.

8. Conclusions

This paper has defined, for the first time, the problem of data survival in UWSNs in the presence of a capable mobile adversary. Besides exposing the problem, we probed a number of interesting issues having to do with the power of the adversary and the degree of data resilience supported by the network. In particular, we introduced a network and a threat model which allow systematic study of a variety of survival/attack strategies. We also considered the use of replication as one means of increasing survival probability of high-value data. Both analytical and simulation results show that proposed techniques offer non-negligible guarantees of data survival. Although we addressed these issue without resorting to encryption, we sketched out how the use of encryption prompts a number of new issues. (This is one of the subjects of our on-going work.)

References

[1] G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable rfid tags via insubvertible encryption. In *CCS '05*, pages 92–101, 2005.

[2] M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. Cryptology ePrint Archive, Report 2004/221, 2004.

[3] M. Bellare and B. S. Yee. Forward-security in private-key cryptography. In *CT-RSA*, pages 1–18, 2003.

[4] V. Berman and B. Mukherjee. Data security in manets using multipath routing and directional transmission. In *IEEE International Conference on Communications (ICC'06)*, pages 2322–2328, 2006.

[5] S. Chessa and P. Maestrini. Dependable and secure data storage and retrieval in mobile, wireless networks. In *DSN 2003*, pages 207–216, 2003.

[6] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik. Catch me (if you can): Data survival in unattended sensor networks. In *http://www.dsi.uniroma1.it/∼dipietro/catchme.pdf*, 2007.

[7] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of rsa functions. In *CRYPTO*, pages 157–172, 1996.

[8] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold dss signatures. In *EUROCRYPT*, pages 354–371, 1996.

[9] V. Gianuzzi. Data replication effectiveness in mobile ad-hoc networks. In *ACM PE-WASUN '04*, pages 17–22, 2004.

[10] T. Hara and S. K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Trans. Mob. Comput.*, 5:1515–1532, 2006.

[11] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: maximizing sensor network data persistence. *SIGCOMM Comput. Commun. Rev.*, 36(4):255–266, 2006.

[12] D. Ma and G. Tsudik. Forward-secure sequential aggregate authentication. In *IEEE Symposium on Research in Security and Privacy, (S&P'07)*, 2007, to appear.

[13] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *PODC*, pages 51–59, 1991.

[14] P. Papadimitratos and Z. Haas. Secure data communication in mobile ad hoc networks. *IEEE JSAC*, 24(2):343–356, 2006.

[15] V. Shoup. Oaep reconsidered. Cryptology ePrint Archive, Report 2000/060, 2000.