Maximizing data survival in Unattended Wireless Sensor Networks against a focused mobile adversary

Roberto Di Pietro[a], Luigi V. Mancini[b], Claudio Soriente[c], Angelo Spognardi[d], Gene Tsudik[c]

[a] *Dipartimento di Matematica, Università degli studi di Roma Tre*
[b] *Dipartimento di Informatica, Università "La Sapienza" Roma*
[c] *Computer Science Department, University of California, Irvine*
[d] *Equipe Planète, INRIA Rhône-Alpes*

Abstract

Some sensor network settings involve disconnected or unattended operation with periodic visits by a mobile sink. An unattended sensor network operating in a hostile environment can collect data that represents a high-value target for the adversary. Since an unattended sensor can not immediately off-load sensed data to a safe external entity (such as a sink), the adversary can easily mount a focused attack aiming to erase or modify target data. To maximize chances of data survival, sensors must collaboratively attempt to mislead the adversary and hide the location, the origin, and the contents of collected data.

In this paper, we focus on applications of well-known security techniques to maximize chances of data survival in unattended sensor networks, where sensed data can not be off-loaded to a sink in real time. Our investigation yields some interesting insights and surprising results. The highlights of our work are: (1) thorough exploration of the data survival challenge, (2) exploration of the design space for possible solutions, (3) construction of several practical and effective techniques, and (4) their evaluation.

Key words: Unattended WSN, data survival, security, mobile adversary, probabilistic analysis.

1. Introduction

In recent years, sensors and sensor networks have been extremely popular in the research community. Much of prior research explored various aspects of *Wireless Sensor Networks* (WSNs), including: system architecture, routing, security, power-awareness and data abstraction. In particular, security issues in WSNs have received a lot of attention. One common assumption in prior WSN

Email addresses: dipietro@mat.uniroma3.it (Roberto Di Pietro), mancini@di.uniroma1.it (Luigi V. Mancini), csorient@ics.uci.edu (Claudio Soriente), spognard@inrialpes.fr (Angelo Spognardi), gts@ics.uci.edu (Gene Tsudik)

security research is that data collection is performed in, or near, real time. In other words, a trusted entity (such as a sink) is assumed to be always present. Individual sensors submit their data to the sink either periodically or based on some external trigger, e.g., a change in the sensed environment or an explicit request by the sink.

Another emerging sensor network type involves sensor mobility and opportunistic connectivity among sensors as well as between sensors and the sink [1, 2, 3]. This concept is similar to Delay Tolerant Networks (DTNs). It is characterized by sensors' inability to communicate with other sensors, for reasons such as: limited transmission ranges, power constraints or signal propagation problems (e.g., line-of-sight limitations or physical obstacles).

In this paper, we focus on WSN scenarios and applications that do not fit into either the real-time data collection model or the opportunistic DTN-like model. We are interested in *sensor networks* where sensors are connected but there is no real-time communication with the sink. We refer to such networks as *Unattended WSNs* or UWSNs. We narrow our scope even further to UWSNs operating in a hostile – or at least untrusted – environment where the adversary has free reign. Specifically, the adversary has one central goal: to prevent certain data collected by sensors from ever reaching the sink. We elaborate on this below.

One example of hostile unattended environment could be a network of nuclear emission sensors deployed in a recalcitrant country (under, say, 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 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.
Quite recently, the U.S. Defense Advanced Research Projects Agency (DARPA) initiated a new research program to develop so-called LANdroids [4]: smart robotic radio relay nodes for battlefield deployment. LANdroid nodes are supposed to be deployed in hostile environment, establish an ad-hoc network, and provide connectivity as well as valuable information for soldiers that would later approach the deployment area. LANdroids might retain valuable information for a long time, until soldiers move close to the network. In the interim, the adversary might attempt to delete or modify that information, without disrupting network operations, so as to remain undetected.

In such settings, the greatest challenge 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 who is aware of the origin(s) of targeted 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. This type of adversary has been studied in the cryptographic literature where it is usually referred to as a *Mobile Adversary* [5]. An entire branch of cryptography, called *Proactive Cryptography* has been dedicated to developing cryptographic techniques (e.g., decryption and digital signatures [6, 7]) that remain secure in the presence of a mobile adversary. Although our adversary models are similar, the UWSN application domain is very different from that in proactive cryptography (as described below), thus motivating radically different solutions.

Scope. This paper represents the very first attempt to develop cryptographic defenses for coping with a focused mobile adversary in UWSNs. However, as becomes clear throughout, this paper **does not** address a number of important problems. This is partly because of space limitations and partly due to the novel nature of the topic and problem at hand. We expect that this paper will result in follow-on investigations on our part as well on the part of the research community.

We also stress that our work is oriented towards sensor networks and is not particularly novel in terms of cryptography. Its novelty stems from applying well-known and accepted cryptographic tools to solving a novel networking problem.

Our Contributions. This paper provides the following contributions:

1. *Problem Exposure:* although some recent work [8] first brought the problem to light, it focused on trivial and intuitive data survival strategies. In contrast, the present work delves much deeper into the problem and constructs effective and efficient countermeasures that achieve our main goal of maximizing data survival in UWSNs in the presence of a powerful mobile adversary.
2. *Novel Techniques & Analysis:* we thoroughly explore the design space of cryptographic solutions and – without resorting to expensive and/or exotic techniques – develop several practical and optimal (or near-optimal) data survival strategies. Our investigation yields some interesting results; for instance, when using public key cryptography, continuously moving data around the network provides the same security of combining the following techniques: moving data just once, plus re-encryption. Further, our evaluations of proposed techniques demonstrate a surprising degree of data survival even when the adversary is very agile and powerful, while the sensor network remains unattended for a relatively long time.

Organization. Section 2 introduces our environment assumptions. Then, Section 3 explores potential data survival strategies for the UWSN, adversarial counter-strategies and a number of design parameters. Section 4 investigates encryption–related issues and parameters. Section 5 presents our analysis. Next, Section 6 overviews relevant prior work. Finally, Section 7 provides a summary and some directions for future work.

2. System Assumptions

In this section we present our assumptions about the sensor network environment and the adversary.

2.1. Network Environment

We envisage a UWSN which operates as follows:

- Sensors are programmed to sense and collect data periodically. There is a fixed global periodicity parameter p denoting the time interval between successive sensing operations.
- Each sensor collects a single unit of data for each interval. In an UWSN composed of n sensors, we say, sensor s_j collects data d_j^r for interval r.
- The network is unattended. There exists a parameter q ($q = v * p$ for some integer v) which denotes the maximum time between successive visits of the sink or collector —we use the term sink from here on to mean both.
- As soon as each sensor off-loads its accumulated data to the sink, it erases its entire storage. Moreover, the sink re-initializes all sensors' secret material upon each visit. In other words, any secret values held by a sensor right before the sink visit are completely independent from those held after the visit.
- 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 nature of the network. Indeed, our results are independent from the underlying routing protocol.
- There are no power constraints. At least initially, we are not concerned with power consumption of various survival 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 the unattended nature of the network, we are considering an emerging kind of sensor network not typically encountered in the research literature.

2.2. Portrait of the Adversary

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

- Compromise power: $\mathcal{ADV}$ is capable of compromising at most k out of n sensors during any single interval. k may be a fixed integer value or a fraction of n —number of sensors in the network. Once $\mathcal{ADV}$ compromises a sensor, and as long as it remains compromised, we assume that $\mathcal{ADV}$ reads

all of its storage and monitors all incoming and outgoing communications. We do not assume that the subset of compromised sensors is clustered or contiguous, i.e., concurrently compromised sensors can be spread through the entire network.

- Compromise Round & Collection Round: for ease of exposition and without loss of generality, we assume that the compromise and collection rounds have the same duration. Moreover, and also without loss of generality, we assume that they are synchronized, i.e., both types of rounds start and end at the same time.

- Network knowledge: $\mathcal{ADV}$ knows the composition and the topology of the network.

- Limited erasure capacity: between any two successive sink visits (within v intervals) $\mathcal{ADV}$ can erase no more than a given number t of measurements from the network. Erasing more than that, raises an alarm on the sink and contradicts $\mathcal{ADV}$'s goal of remaining undetected.[1]

- No interference: except for the above, $\mathcal{ADV}$ does not interfere with communications of any sensor and does not *modify* any other data sensed by – or stored on – sensors it compromises; this assumption as well will be re-considered later.

- Atomic movement: $\mathcal{ADV}$ moves in one fell swoop, i.e., at the end of each interval it selects at most k sensors to compromise in the next interval and migrates to them in one monolithic step. Note that the two sets of compromised sensors may intersect or even be the same. Our assumptions about adversary movements are similar to the one in [9].

- Stealthy operation: $\mathcal{ADV}$'s movements between intervals are unpredictable and untraceable. As it moves from one set of k sensors to the next, $\mathcal{ADV}$ leaves no trace behind. This implies that a compromised sensor released by $\mathcal{ADV}$ is fully operational

With reference to the last item, we assume that $\mathcal{ADV}$ does not modify any data it encounters as it compromises sensors. It also does not inject any data of its own. An important consequence is that, in this paper, we are not addressing the data authenticity problem. We are concerned only with data survival, which motivates hiding: (1) data origin, (2) data content, and (3) time of data collection. The reason for hiding these three values is apparent – we want to minimize information available to $\mathcal{ADV}$ as it roams around the UWSN looking for the target data.

We distinguish between a **proactive** and a **reactive** adversary. The latter is assumed to be dormant (inactive) until it gets a signal that certain data must

[1] Whereas, a few missing reports might be considered by the sink as to be a consequence of sensor malfunctioning.

be erased. As soon as this happens, $\mathcal{ADV}$ *reacts* and starts compromising, in each round, up to k sensors. In contrast, a *proactive* $\mathcal{ADV}$ roams the network ahead of time, waiting for a signal to erase certain data. The reason for this distinction is discussed later on in the paper.

Finally, in Table 1 we summarize the notation used in the rest of the paper. We use the terms *round* and *interval* interchangeably, to denote the time between successive sensor measurements.

n	size of the UWSN
i, j	sensor indices
s_i	sensor i
r, r'	round/interval indices
d_i^r	data collected by sensor i at interval r
$S(d_i^r, r')$	sensor hosting d_i^r at round $r' > r$
K_i^r	key used by sensor s_i at round r
$U^{r'}$	set of data items undecipherable by $\mathcal{ADV}$ in round r'
v	number of rounds between successive sink visits
C_r	set of compromised sensors at round r
k	maximum size of C_r; assumed constant

Table 1: Notation Summary

3. Strategies and Design Parameters

In this section we introduce possible strategies adopted by the adversary to reach its goals and the countermeasures taken up by the network to ensure data survival. We also survey other design parameters such as encryption, authentication and replication.

3.1. Survival and Attack Strategies

Our main goal is to maximize survival probability for data collected by sensor s_i at interval r (that is, d_i^r). Survival means that this data is eventually delivered to the sink. At round r $\mathcal{ADV}$ learns from an external signal which data it has to erase, namely, it learns both s_i and r. Unfortunately for us, $\mathcal{ADV}$ does not reveal the data it is interested in *erasing*; thus, we know neither of these values, except that $1 \leq i \leq n$ and $0 \leq r \leq v$. We must therefore assume that **all data** is potentially targeted by $\mathcal{ADV}$.

Focusing strictly on non-cryptographic techniques [8] considered two intuitive data survival strategies:

MOVE-ONCE: at every round r, each sensor s_j collects data d_j^r, randomly picks a sensor —that is going to become $S(d_j^r, r+1)$— and sends d_j^r to it. Thereafter, d_j^r remains at its new "home" until the next sink visit.

KEEP-MOVING: at each round r each sensor moves each hosted data item separately, i.e., for each data item that it stores (and collects), it picks a random

sensor and moves there the stored item. As we show later in the paper, this strategy does not significantly increase data survival chances.

Whichever survival strategy is used, one must assume that $\mathcal{ADV}$ is aware of it. Knowing the survival strategy lets $\mathcal{ADV}$ pick a counter-strategy that maximizes chances of deleting the target data. [8] considered several counter-strategies that, given a sufficiently large v (number of rounds between sink visits), guaranteed that $\mathcal{ADV}$ wins the game as long as data is kept as cleartext. These survival strategies vary only as far as exactly how many rounds it takes $\mathcal{ADV}$ to win.

The use of encryption allows us to hide the origin, the time of collection and the content of sensed data. If $\mathcal{ADV}$ can not recognize target data, former attack strategies no longer apply and $\mathcal{ADV}$ is forced to erase data blindly, i.e., to guess which ciphertext hides the target data. In the analysis below, we use $U^{r'}$ to denote the set of all encrypted data items that $\mathcal{ADV}$ can not decrypt at round r'. We stress that message eavesdropping or interception does not affect our security analysis, since we focus only on the indistinguishability of messages. The greater the size of $U^{r'}$, the higher the probability that the target data will persist until the next sink visit. In particular, given t possible erasures, $\mathcal{ADV}$ has probability $\frac{t}{|U^{r'}|}$ of succeeding. The survival strategy aims to increase the size of $U^{r'}$. Whereas, $\mathcal{ADV}$'s counter-strategy is to roam the network and learn as much information as possible in order to maximize the chances of finding and erasing d_i^r. To this end, $\mathcal{ADV}$'s goal is to limit the growth of $|U^{r'}|$ and if possible, even to decrease it.

3.2. Design Parameters

Encryption. In the context of this paper, the most important issue is encryption. If sensors use encryption in conjunction to hiding data location (by moving data around), they can hide not only the contents of collected data but also the identity of the sensor that collected it as well as the round identifier (i.e., the time of collection). Use of encryption is a natural choice; however, it comes with certain non-negligible costs, such as key management and the overhead due to cryptographic operations. Encryption also motivates certain assumptions and technicalities which we discuss in the rest of this paper.

Authentication. Another important issue is authentication, i.e., whether the sink can establish with certainty both data integrity and data origin authenticity. As mentioned in Section 2.2, we are not dealing with authentication in this paper. More concretely, we assume that all data is encrypted using Plaintext-Aware Encryption [10] whereby any modification (without knowledge of the secret key) will produce gibberish upon attempted decryption.

We expect that follow-on works will address richer adversary model which allows $\mathcal{ADV}$ to modify existing data and/or inject fake data into compromised sensors.

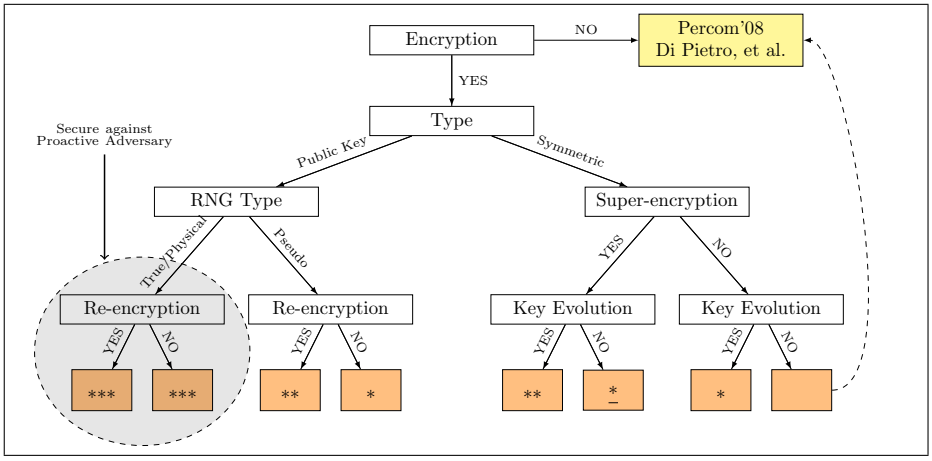

Figure 1: Decision Tree

Replication. The final parameter we consider 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. Replication of cleartext data was previously studied in [8]. Although replication of encrypted data is more effective than cleartext replication for defending against our focused mobile adversary, we consider replication an independent issue and do not discuss it further in this paper.

4. Encryption Parameters and Features

We are primarily concerned with encryption as a means of hiding the origin and time of collection (and to a lesser extent, the contents) of sensed data. Furthermore, we assume that, regardless of the encryption details, encryption is always randomized [11], which (informally) means that given two encryptions under the same key, it is unfeasible to determine whether the corresponding plaintexts are the same.

We now discuss encryption features and parameters. To simplify the discussion, we show the "decision tree" in Figure 1 where leaves represent specific techniques. Note that the *asterisks* (*) associated with the leaves provide an intuitive measure of the quality of the data survival strategy represented: the more asterisks, the bigger the set $|U^{r'}|$.[2] Justifications for the rankings – as well

[2]However, note that $\underline{*}$ has a special meaning: the exact quality depends on the relationship between r' and k. In particular, for $r' < n/k$, it is better than that provided by a single asterisk. On the other hand, if $r' > n/k$ the quality is lower than that of all other techniques.

as the meaning of the dashed arrow – are clarified below, in the course of our discussion.

As usual, the choice of public key or symmetric (shared key) algorithm is the main variable when introducing encryption. We remain agnostic with respect to this choice and consider both cases.

4.1. Symmetric Encryption

Our construction with conventional encryption is straightforward:

1. Each sensor s_j shares a distinct initial key K_j^0 with the sink.
2. At round r, sensor s_j senses data unit d_j^r and encrypts it to produce $E_j^r = E(K_j^r, d_j^r, r, s_j, etc.)$.
3. Finally, s_j picks a random destination sensor s_l and sends E_j^r to it.
4. If KEEP-MOVING strategy is used, s_j sends each stored data item to random destination sensors.

Since symmetric encryption is inherently invertible, we need to worry about what happens when $\mathcal{ADV}$ compromises sensor s_i which originated target data d_i^r. By the time $\mathcal{ADV}$ compromises s_i, d_i^r is off-loaded to another sensor $S(d_i^r, r')$ (assuming r' is the current round). However, if s_i's key K_i^0 does not change, as soon as $\mathcal{ADV}$ learns this key, it becomes capable of recognizing d_i^r – by decrypting its cyphertext with K_i^0– whenever, at some future round r', $\mathcal{ADV}$ compromises $S(d_i^r, r')$. We therefore claim that the security offered by symmetric encryption is the same as in the case of not using encryption. This is why Figure 1 shows a dashed arrow from the right-most leaf to top box, denoting the essential equivalence of using a constant (non-evolving) shared key for encryption and not using encryption at all.

To cope with sensor compromise, we need a property commonly referred to as *Forward Secrecy* [12] which, in our case, can be easily obtained by *evolving* the key in each round, using any suitable one-way function $OWF(\circ)$. Concretely, we introduce an additional step between steps 2 and 3 described above:

2(a) s_j computes $K_j^{r+1} = OWF(K_j^r)$ where $OWF(\circ)$ is a cryptographically suitable one-way function, such as SHA-2. Then, K_j^r is deleted.

A minor issue is how the sink would decrypt data, i.e., how it would determine which decryption key to use. This actually does not pose a problem since the sink has all initial keys of the form K_j^0. It can attempt to decrypt a ciphertext using all $n * v$ possible keys. While this might seem excessive, we point out that symmetric encryption is very inexpensive and the sink is assumed to have no computational constraints.

Unfortunately, symmetric encryption offers security only against a *reactive* $\mathcal{ADV}$. To see this, consider a *proactive* $\mathcal{ADV}$ who roams the network for $\lceil \frac{n}{k} \rceil$ rounds prior to receiving a signal that, at round $r = \left(\lceil \frac{n}{k} \rceil + 1\right)$ sensor s_i generated target data d_i^r. $\mathcal{ADV}$ has already "visited" all n sensors in previous $\lceil \frac{n}{k} \rceil$ rounds. Therefore, it is able to derive each symmetric keys used by every sensor

in round r.[3] At this point, $\mathcal{ADV}$ just needs to find the current location of E_i^r and erase it, before the sink visits the network. Of course, this is the worst-case scenario where $\mathcal{ADV}$ is guaranteed to win. More generally, a proactive $\mathcal{ADV}$ does not necessarily have the luxury of $\lceil \frac{n}{k} \rceil$ rounds before receiving the signal.

One possible, but limited, measure against a proactive $\mathcal{ADV}$ is *super-encryption*, i.e., further encryption of already encrypted (E_i^r) data by the host sensor. The motivation is to address the case when, before getting the signal, $\mathcal{ADV}$ compromises only either the originator s_i or the host sensor $s_j = S(d_i^r, r+1)$. If so, at round $r+1$, d_i^r is stored at s_j as:

$$E(K_j^r, E_i^r) = E(K_j^r,\ E(K_i^r, d_i^r, r, s_i, etc.))$$

To recognize d_i^r, $\mathcal{ADV}$ needs to decrypt both layers, which is impossible without knowledge of both K_i^r and K_j^r. Note that the above implicitly refers to the MOVE-ONCE strategy. It is easy to extend super-encryption to the KEEP-MOVING.

In summary, symmetric super-encryption offers only limited help against a proactive $\mathcal{ADV}$. Its exact effectiveness is assessed below in Section 5.2.3.

4.2. Public Key Encryption

In the past, public key encryption was often avoided in the sensor network security literature since its higher cost was viewed as a poor match for low-end sensors. However, due to recent advances, public key encryption is becoming more appealing [13]. Furthermore, as we show below, use of public key encryption offers a level of security unattainable with symmetric encryption. The base case for public key encryption has the following features:

1. The sink has a long-term public key, PK_{sink}, known to all sensors.
2. At round r, sensor s_j collects data d_j^r and encrypts it to produce $E_j^r = E(PK_{sink}, K_j^r, d_j^r, r, s_j, etc.)$ [4].
3. As in the symmetric case, s_j picks a random destination sensor s_l and sends E_j^r to it.
4. As before, with KEEP-MOVING, all stored data items are sent to random sensors as well.

Note that sensors have no secret (private) keys of their own – they merely use the sink's public key to encrypt data. However, even this simple approach results in $\mathcal{ADV}$ being unable to distinguish target data among all other encrypted data it finds on compromised sensors.

Since $\mathcal{ADV}$ does not know the sink's decryption key (SK_{sink}), the only way it can attempt to detect target data is by trying to encrypt its duplicate[5] under the sink's public key, PK_{sink}.

---

[3] Since knowing a sensor's key for a given round allows it to derive the same sensor's keys for all subsequent rounds.

[4] K_j^r is the random number provided at round r by the random number generator of s_j

[5] We assume that $\mathcal{ADV}$ can easily guess the actual sensed data d_i^r.

This is where our randomized encryption assumption comes in handy. With randomized encryption, each encryption operation involves generating a one-time random number and *folding* it into the plaintext (e.g., as in OAEP+ [11]) such that, without knowledge of that unique random number, it is computationally unfeasible to re-create the same ciphertext. Consequently, $\mathcal{ADV}$ is unable to distinguish among different encrypted values it finds on compromised sensors.

There is, however, a crucial security distinction based on the source of random numbers. If random numbers are obtained from a strong (fully or, at least partially, physical) source of randomness, then we can achieve the maximum level of security — we also assume that the random numbers are sufficiently long to make exhaustive guessing computationally unfeasible, i.e., at least 128 bits. To argue this claim informally, consider that a true random number generator (TRNG) generates information-theoretically independent values. That is, given an arbitrarily long sequence of consecutive TRNG-generated numbers, removing any one number from the sequence makes any guess of the missing number equally likely.

On one hand, the only way for proactive $\mathcal{ADV}$ to recognize E_i^r is if the compromised set C_r includes s_i at the exact round r when $\mathcal{ADV}$ receives the signal. This is, indeed, the highest level of security we can possibly hope to achieve —in other words, $\mathcal{ADV}$ can win only due to dumb luck. If, on the other hand, *random* numbers are obtained from a pseudo-random number generator (PRNG), the resulting security is equivalent to that of the symmetric encryption case with key evolution. This is because a typical PRNG produces numbers by starting with a seed value and repeatedly applying a suitably strong one-way function. Hence, it is functionally equivalent to the key-evolution feature described in the previous section. This leads us to conclude that: *if sensors have no real source of randomness, there is no reason to use public key encryption, since it costs more than symmetric encryption and does not offer better security.* However, an interesting feature of public key encryption is that some techniques (e.g., [14]) allow what is referred to as *re-encryption*. In our context, this means that a sensor which receives an 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. Two advantages of using re-encryption are:

- It does not require multiple decryption operations to obtain the cleartext (unlike super-encryption).
- It does not cause ciphertext expansion.

The use of re-encryption is beneficial considering that $\mathcal{ADV}$, as it breaks into sensors, might attempt to copy and remember encrypted values that sensors in C_r generate, encrypt and off-load during the same round: that way it could detect them later. If all sensors re-encrypt all values they receive from others, $\mathcal{ADV}$ is placed at a further disadvantage. Moreover, if all data stored on a sensor are re-encrypted at every round, we further reduce the chances of $\mathcal{ADV}$ to find out the data it is looking for. We demonstrate the effects of the above strategy in Section 5.3.

Unfortunately, re-encryption imposes a peculiar limitation: it precludes the use of hybrid (envelope) encryption. By *hybrid* we mean the way that public key encryption is typically used in practice: a one-time symmetric key is generated and encrypted using the public key, and the bulk data is then encrypted using the said symmetric key. Hybrid encryption produces a two-part ciphertext: public and symmetric. Re-encryption can be applied to the former but **not** to the latter. (Of course, super-encryption can be used instead, but that would negate all advantages of using re-encryption.) Therefore, re-encryption is useful only if data (d_i^r) is sufficiently short to fit into a single public key encryption block.[6]

5. Analysis and Discussion

In this section we analyze the cryptographic techniques outlined above. In doing so, we explore the leaves of the decision tree in Figure 1, and provide survival probability of the target data for both MOVE-ONCE and KEEP-MOVING network defense strategies. We start describing the experimental testbed we used to validate all our theoretical results.

5.1. Experimental testbed

We developed a UWSN simulator able to recreate the different combinations of adversarial and defense strategies. For every scenario, we run the simulation 100 times: a simulation lasts for 50 rounds. In each round sensors sense, encrypt and randomly exchange ciphertext according to the defense strategy, while $\mathcal{ADV}$ compromises, acquires keys, and attempts to decrypt the captured ciphertexts. For each round, we computed the average number of ciphertexts that $\mathcal{ADV}$ was unable to distinguish from a target data sensed at the first round. Finally, the simulator was developed in C++ and we used the Mersenne Twister [15] as pseudorandom number generator.

5.2. Symmetric Encryption

We assess the effectiveness of symmetric encryption along with features, such as *key evolution* and *super-encryption*.

5.2.1. Plain Encryption

With MOVE-ONCE, $\mathcal{ADV}$ wins in at most $\lceil \frac{n}{k} \rceil$ rounds with the following counter-strategy. In the first round after the occurrence of target data (round $r+1$), $\mathcal{ADV}$ chooses C_{r+1} such that $s_i \in C_{r+1}$. This way it learns K_i^0 – the key used by s_i to encrypt d_i^r. It then uses K_i^0 to try to decrypt all ciphertexts found on all currently compromised sensors. It thus takes $\mathcal{ADV}$ at most $\lceil \frac{n}{k} \rceil$ rounds

[6] In principle, one could use public key encryption over multiple blocks of data and re-encrypt each ciphertext block separately; however, we consider this to be a very particularly unappealing approach.

to visit all sensors and find d_i^r. In the following, without losing of generality, we will assume that $\frac{n}{k}$ is an integer.

With KEEP-MOVING, $\mathcal{ADV}$ wins in $\frac{n}{2k}$ rounds, on average [8]. It learns K_i^0 at round $r+1$ (by choosing C_{r+1} such that $s_i \in C_{r+1}$), and deletes d_i^r as soon as it is found in one of the compromised sensors.

In summary, if plain symmetric encryption is used and the number of rounds between successive sink visit exceeds $\frac{n}{k}$, $\mathcal{ADV}$ can always recognize and, once found, erase target data, regardless of the data moving strategy. Furthermore, symmetric encryption alone offers no advantage over cleartext survival strategies.

5.2.2. Key Evolution

Adding a simple per round *key evolution* feature, results in the continual increase of $U^{r'}$ for the first $r+\frac{n}{k}$ rounds assuming that $\mathcal{ADV}$ starts compromising sensors at round $r+1$. $U^{r'}$ then remains constant for all subsequent rounds.

To justify this claim, recall our discussion in Section 4.1. At each round, every sensor's key is computed by applying $OWF(\circ)$ to the key used at the previous round. If $\mathcal{ADV}$ initially compromises a generic sensor s_j at round r, it learns K_j^r and can thus compute $K_j^{r'}$ for any $r' > r$. However, it can not compute any key used by s_j prior to round r. In particular, if a reactive $\mathcal{ADV}$ decides to erase target data at round r and starts compromising sensors at round $r+1$, it can not learn K_i^r, even if C_{r+1} is chosen such that $s_i \in C_{r+1}$.

Consequently, any data item encrypted with a key that $\mathcal{ADV}$ can not compute at round r', is an element of $U^{r'}$. If $\mathcal{ADV}$ starts collecting keys at round $r+1$, then after $\frac{n}{k}$ rounds, it learns one key for every sensor in the network and $|U^{r'}|$ stops growing for all $r' > r + \frac{n}{k}$. Then, the size of $U^{r'}$ will be

$$|U^{r'}| = (r'+1)\, n - \sum_{i=1}^{r'-r} n\ \frac{ik}{n} \tag{1}$$

until $r' < r + \frac{n}{k}$, while $|U^{r'}| = |U^{r+\frac{n}{k}-1}|$ afterward, as shown in Figure 2.

5.2.3. Super-Encryption

Super-encryption entails each host sensor encrypting (already-encrypted) data items it receives. For example, d_i^r originally sensed by s_i and sent to s_j at round r, and then sent to s_l at round $r+1$, will be stored by s_l as: $E(K_l^{r+1}, E(K_j^r, E_i^r))$.

With MOVE-ONCE, $\mathcal{ADV}$ can find and erase target data in at most $\frac{n}{k}$ rounds – as in Section 5.2.1 – with the following counter-strategy:

(1) Assume d_i^r is stored at s_j as $E(K_j^r, E_i^r)$.

(2) $\mathcal{ADV}$ chooses C_{r+1} such that $s_i \in C_{r+1}$; this way it learns K_i^r.

(3) For each compromised sensor in C_{r+1} and for each ciphertext found therein, $\mathcal{ADV}$ first attempts decryption with the sensor's current key, and then attempts decrypting the output with K_i^r: if d_i^r is found, $\mathcal{ADV}$ deletes it. Note that it takes at most $\frac{n}{k}$ rounds to traverse all sensors of the network.

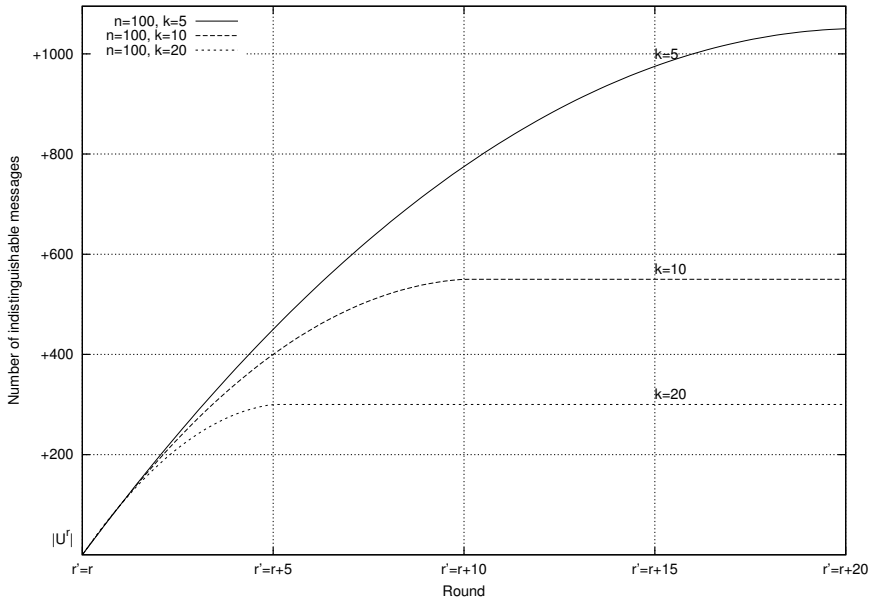

Figure 2: Growth of $|U^{r'}|$ with KEEP-MOVING and symmetric key encryption with key evolution. Once $\mathcal{ADV}$ has visited all sensor, the set of indistinguishable ciphertexts stops growing.

We now look at the KEEP-MOVING strategy. Let $\overline{C}_{r'}$ be the set of sensors **not compromised** within the first r' rounds. At any round r', all ciphertexts encrypted at least once by any sensor in $\overline{C}_{r'}$ are undecipherable by $\mathcal{ADV}$.

The probability that d^j (any data item collected at round $j \leq r'$), *at each round*, is encrypted and then super-encrypted with a key acquired by $\mathcal{ADV}$ until round r' is $\left(\frac{(r'-r)k}{n}\right)^{r'-j+2}$. Thus, the probability of d^j being encrypted or super-encrypted at least once with a key unknown to $\mathcal{ADV}$ is $1-\left(\frac{(r'-r)k}{n}\right)^{r'-j+2}$. For data obtained after round $r + \frac{n}{k}$, that probability is 0, since $\mathcal{ADV}$ has compromised all the sensors once and corrupted all the possible keys. Then, with n new data items per round, for $r' \leq r$, $|U^{r'}|$ will be simply

$$|U^{r'}| = (r'+1)n$$

while it will be:

$$|U^{r'}| = \sum_{j=0}^{r'} n\left(1 - \left(\frac{(r'-r)k}{n}\right)^{r'-j+2}\right) \tag{2}$$

for $r < r' \leq r + \frac{n}{k}$ and simply

$$|U^{r'}| = 0$$

for $r' > r + \frac{n}{k}$.

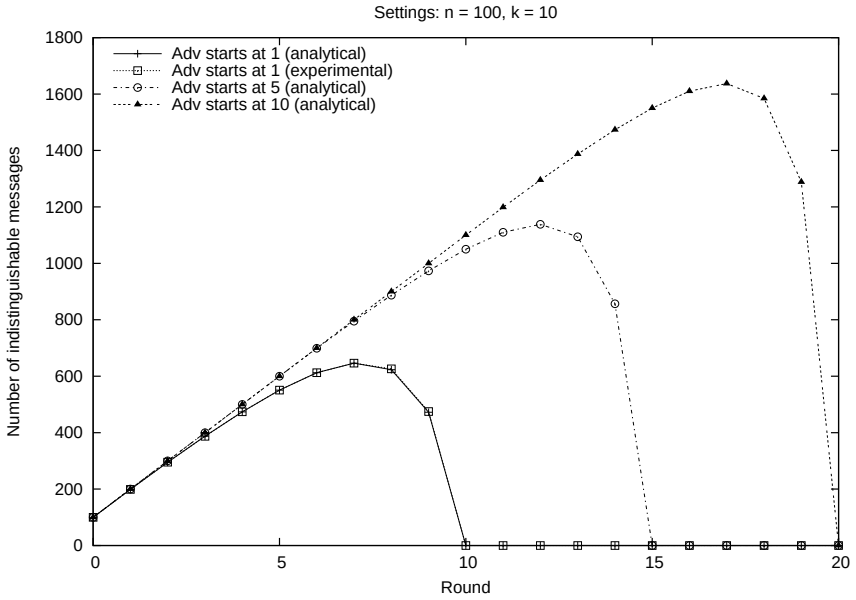

Figure 3: Super-encryption, with $\mathcal{ADV}$ starts at different rounds. Once $\mathcal{ADV}$ has visited all sensors, all the data can be decrypted.

Equation (2) is highly influenced by $r' - r$, namely the number of rounds that $\mathcal{ADV}$ waited before starting compromising the first set of sensors. This implies that the size of $|U^{r'}|$ depends on the number of already collected data at round r. Figure 3 shows the difference of $|U^{r'}|$ between two different adversarial behaviors. In one case, $\mathcal{ADV}$ starts compromising sensors at round 1, that is when sensors have collected only one data item. In the second case, $\mathcal{ADV}$ starts compromise activities at some later rounds, when sensors have several data items in their storage, namely round 5 and 10. In the figure, the first analytical data plot is compared with the simulation results, to endorse the validity of the derived formulas.

Super-encryption has a small caveat: as described in Section 3.2, data is assumed to be encrypted using Plaintext-Aware Encryption (PAE) [10] thereby causing ciphertext expansion. Thus, for each layer of encryption, ciphertext grows by a constant number (e.g., 128) of bits. While this is not an issue for MOVE-ONCE, with KEEP-MOVING ciphertext size grows linearly with the number of moves (rounds). Also, ciphertext size in KEEP-MOVING leaks number of moves that a particular ciphertext has gone through.

5.2.4. Key Evolution and Super-encryption

We now look at the variant combining Key Evolution and Super-Encryption. With MOVE-ONCE, at round r', $\mathcal{ADV}$ can not decrypt ciphertexts in the following three categories:

1. Encrypted and super-encrypted with unknown keys:

$$\left((n-(r'-r)k)\frac{n-(r'-r)k}{n}\right)$$

2. Encrypted with a known key and super-encrypted with an unknown key:

$$\left((r'-r)k\frac{n-(r'-r)k}{n}\right)$$

3. Encrypted with an unknown key and super-encrypted with a known key:

$$\left((n-(r'-r)k)\frac{(r'-r)k}{n}\right)$$

At each round after r, from the latter category $\mathcal{ADV}$ can exclude k ciphertext (on average) to be its target data: they are received by the compromised nodes and, then, can be recognized as ciphertext of data sensed at a round later than r. Combining the previous equations, we can state that

$$|U^{r'}| = (r+1)n + \sum_{i=r+1}^{r'}\left(n-k-\frac{(i-1-r)k^2}{n}\right) \tag{3}$$

when $r<r'<r+\frac{n}{k}$ and stabilizes to $|U^{r+\frac{n}{k}-1}|$, when $r' \geq r+\frac{n}{k}$, i.e. when $\mathcal{ADV}$ visited all the nodes at least once.

As a matter of fact, there is an interesting subtlety involved in estimating growth of $U^{r'}$ above: $\mathcal{ADV}$ can keep track of all the ciphertexts found on a sensor storage at any round $r' > r$. If it evaluates the difference of the storage content between two subsequent visits to the same sensor, it will be able to distinguish the newer ciphertexts from the older ones. Equation (2), in fact, is valid only if sensors super-encrypt at each round all the ciphertexts in their storage. If super-encryption is applied only to the received ciphertexts, its effects will be highly reduced and the growth of $U^{r'}$ will be the same than with the KEEP-MOVING strategy without super-encryption of Equation (1).

With super-encryption and KEEP-MOVING, things change: a ciphertext is not in $U^{r'}$ if and only if all keys used to encrypt and super-encrypt it are known to $\mathcal{ADV}$. The number of ciphertexts $\mathcal{ADV}$ can decrypt is given by the following recurrence equation:

$$W_{r'} = \frac{(r'-r)^2k^2}{n} + \frac{(r'-r)k}{n}W_{r'-1},\ W_0 = 0$$

while the total number of ciphertexts is $n(r'+1)$. Hence, combining these two results, we conclude that with the KEEP-MOVING strategy $|U^{r'}|$ grows as:

$$|U^{r'}| = \begin{cases} (r'+1)n - W^{r'} & \text{if } r < r' < r+\frac{n}{k} \\ |U^{r+\frac{n}{k}-1}| & \text{if } r' \geq r+\frac{n}{k} \end{cases} \tag{4}$$

5.3. Public Key Encryption

We now turn to the public key setting outlined in Section 4.2. Without focusing on a specific public key cryptosystem, we investigate several features of public key cryptography conducive to data survival. As mentioned in Section 4.2, one important issue is the source of randomness: whether sensors have true random number generators (TRNGs) or pseudo-random number generators (PRNGs). As argued in Section 4.2 and shown in Figure 1, public key encryption with a PRNG offers the same data survival probability as symmetric encryption with key evolution. Also, PRNG-based public key encryption with re-encryption offers the same data survival probability as symmetric encryption with key evolution and super-encryption. We thus focus on effectiveness of public key encryption with TRNGs, both with and without re-encryption.

5.3.1. TRNG-based Scheme

If each sensor has a TRNG, $\mathcal{ADV}$ can only distinguish ciphertexts produced by sensors compromised within a given round. Further, $\mathcal{ADV}$ can recognize the same ciphertexts after they are moved elsewhere, if it encounters them later. $\mathcal{ADV}$ can not distinguish other ciphertexts since it is computationally unfeasible to learn random values generated by other sensors, whether previously compromised or not.

If MOVE-ONCE is employed, at round $r+1$ (and thereafter) any ciphertext produced at round r (E_r^j) is located at sensor $S(d_j^r, r+1)$. But, with the same technique seen in the above section, $\mathcal{ADV}$ can distinguish, between two subsequent visits to the same sensor, the older ciphertexts from the newer ones. Then, with the simple public key encryption, the size of $U^{r'}$ has the same growth of Equation (1).

With KEEP-MOVING, instead, $\mathcal{ADV}$ can not determine whether ciphertexts received by compromised sensors have been produced during the current round, i.e., $U^{r'}$ grows every round by only $n-k$ ciphertexts on average. Thus,

$$|U^{r'}| = (r+1)n + (r'-r)(n-k) \tag{5}$$

$$\text{and} \quad |U^{r'}| = (r'+1)(n-k) \tag{6}$$

for a reactive and a proactive $\mathcal{ADV}$, respectively.

5.3.2. TRNG Scheme with Re-Encryption

Combining TRNG-equipped sensors with re-encryption yields a somewhat stronger outcome.

With MOVE-ONCE, each ciphertext is re-encrypted by the randomly selected host sensor. Re-encryption applyed only to the received ciphertexts is not really effective, since $\mathcal{ADV}$ can easily distinguish the ciphertexts between two subsequent visits to the same node: in this case, Equation (1) is again the growth of the set $U^{r'}$.

Re-encryption applyed at each round to all the ciphertexts stored on a sensor makes impossible for $\mathcal{ADV}$ to evaluate the differnce between the storage of the

same sensor in a subsequent round. Indeed, any ciphertext not directly observed by $\mathcal{ADV}$ will be modified and rendered indistinguishable. In this case, $\mathcal{ADV}$ is able to distinguish only the ciphertexts received by the sensors of $C_{r'}$, since each time it moves to another set of sensors, it finds a completely different set of ciphertexts. Thus, the growth of $U^{r'}$ is:

$$|U^{r'}| = (r' + 1)n - k$$

KEEP-MOVING involves re-encryption of all ciphertexts upon every move. At each round, $\mathcal{ADV}$ can only keep track of ciphertexts produced by and exchanged with the set of sensors it currently compromised. At round r', $\mathcal{ADV}$ can track only k ciphertexts obtained by sensors in $C_{r'}$ and sent to sensors in the same set, namely on average, only $k \cdot \frac{k}{n}$. In each subsequent round, on average, a ciphertext tracked by $\mathcal{ADV}$ in the previous round will be still trackable with probability $\frac{k}{n}$. We thus have

$$|U^{r'}| = (r' + 1)n - \sum_{i=r+1}^{r'} \frac{k^{i+1}}{n^i}$$

for both a reactive and a proactive $\mathcal{ADV}$.

5.4. Comparison

Figure 5.4 plots the size of $U^{r'}$ yielded by different strategies discussed thus far. It assumes a UWSN with 100 sensors where $\mathcal{ADV}$ can compromise at most $k = 10$ sensors per round. It clearly shows that, with public key encryption and a TRNG, $|U^{r'}|$ grows at constant rate at every round, even after $\mathcal{ADV}$ compromises each sensor at least once. On the other hand, with symmetric key evolution (with or without super-encryption), $\mathcal{ADV}$ stops the growth of $|U^{r'}|$ after $\frac{n}{k}$ rounds, having compromised each sensor at least once. KEEP-MOVING performs slightly better than MOVE-ONCE as the latter strategy moves only data collected during current round, thus $\mathcal{ADV}$ can be sure that any data moving after round r is not the target one.

The results reflected by our analysis can be summarized as follows:

- Unlike the case of cleartext data migration explored in [8], when (any) encryption is used, KEEP-MOVING offers very little advantage over MOVE-ONCE.
- With re-encryption, the difference between MOVE-ONCE and KEEP-MOVING becomes even negligible. Since KEEP-MOVING strategy requires more overhead, its adoption is desirable only when energy consumption is not a main issue.
- Public key encryption with PRNG and symmetric key encryption with key evolution, offer equivalent data survival chances, regardless of the data moving strategy. Each is secure against a reactive (but not a proactive) $\mathcal{ADV}$. Consequently, unless scalability of key management for the sink is

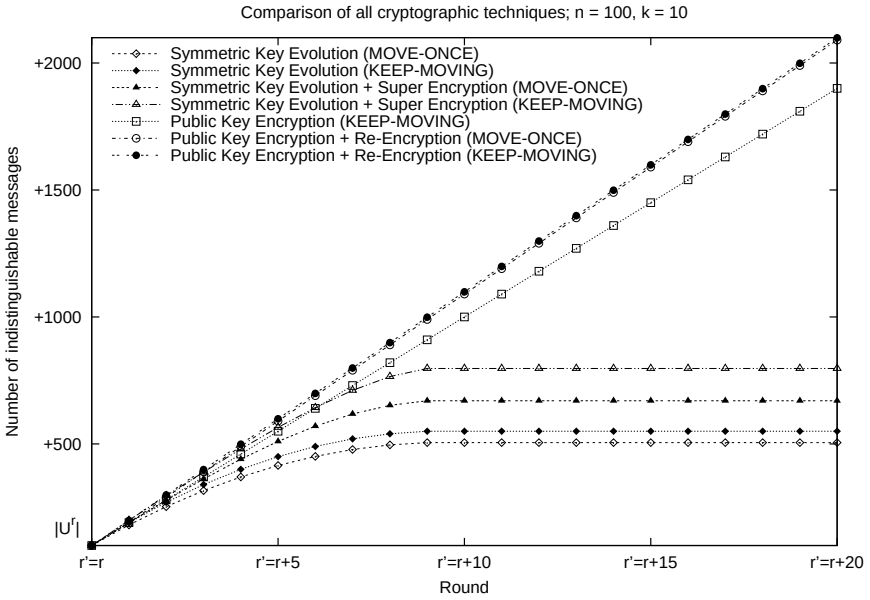

Figure 4: Comparison

an important issue, there is no reason to use public key encryption with PRNG-equipped sensors.

- In presence of a proactive adversary, data survival can be assured only if sensors are required use public key encryption and have TRNG.
- Re-encryption is a useful tool, however, it limits sensors to use "pure" public key encryption, i.e., no hybrid/envelope encryption is possible.

5.5. Invasive $\mathcal{ADV}$

The previous analysis is based on the assumption that $\mathcal{ADV}$ does not interfere with sensors behavior. As mentioned in Section 2.2, we do not address the problem of authenticity of sensed data, then we do not consider the possibility that $\mathcal{ADV}$ modifies in any way data found in sensor storage. Nevertheless, we can relax the non interference assumption, and let $\mathcal{ADV}$ slightly influence sensor behavior. Indeed, if $\mathcal{ADV}$ wants to remain undetected, it is forced not to change much in the way sensors handle sensed and received data. For example, $\mathcal{ADV}$ might change compromised sensors code so that, even if the network strategy is KEEP-MOVING, compromised sensor retain received data (as in the MOVE-ONCE strategy). This change would not be detected by the sink, unless super-encryption is used[7]. With any other crypto technique, the sink can not

[7] Recall that with super-encryption, ciphertext size provides a hint on the number of sensors that added a layer of encryption.

tell the number of sensors that processed a given ciphertext, so it can not distinguish between MOVE-ONCE and KEEP-MOVING. Nevertheless, our analysis show that KEEP-MOVING provides little advantage compared to MOVE-ONCE, so $\mathcal{ADV}$ does not gain much in changing corrupted sensor' strategy. Moreover, a simple analysis on the number of data units stored by each sensor, would reveal to the sink that a part of the network is using a strategy different to the expected one.

$\mathcal{ADV}$ can also change sensor behavior so that each compromised sensor sends its data to other compromised peers. This way, re-encryption is not effective for data generated at compromised sensors. In particular, if compromised sensors exchange messages only among them, the size of $|U^{r'}|$ becomes:

$$|U^{r'}| = (r+1)n + (r'-r)\,(n-2k)$$
$$\text{and} \quad |U^{r'}| = (r'+1)\,(n-2k)$$

with the MOVE-ONCE strategy, for a reactive and a proactive $\mathcal{ADV}$, respectively.

If we consider the KEEP-MOVING strategy, the size of $|U^{r'}|$ becomes:

$$|U^{r'}| = (r+1)n + (r'-r)(n-k)$$
$$\text{and} \quad |U^{r'}| = (r'+1)(n-k)$$

for a reactive and a proactive $\mathcal{ADV}$, respectively.

A more invasive adversary would be easily detected by the sink. For example, if $\mathcal{ADV}$ forces to use arbitrary keys, the sink would detect the anomaly when it visits the network to collect sensed data.

6. Related Work

The problem of data availability in MANETs has been extensively studied, in the relatively benign context of communication faults and network partitions. This thread of works aims to preserve data availability to any MANET node, even when the network is fragmented.
Specifically, Hara, et al. [16] introduced simple yet effective algorithms to replicate data in MANETs, such that, a node in a disconnected partition can access any required 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 technique for location management to guarantee that nodes access the closest data replica. The authors do assume node compromise.

In another related result, Giannuzzi, et al. [17] studied data availability through replication when an ad hoc network is partitioned. This work shows that the probability of accessing certain data is dependent not only on the number of its replicas but also on the network density as well as on the nodes' transmission radius. Neither this work takes into account the possibility of node compromise.

The work of Chessa, et al. [18] introduced a distributed data storage approach for mobile wireless networks, based on the peer-to-peer paradigm. This distributed storage provides support to create and share files under a write-once model, and also ensures data confidentiality and dependability by encoding files in a Redundant Residue Number System (RRNS). Unfortunately, confidentiality is broken as soon as one node is corrupted and the adversary learns the moduli used to encode data under the RRNS.

Finally, some results (e.g., [19, 20]) leverage the existence of multiple paths between end-nodes to statistically improve data confidentiality and data availability in hostile MANET environments, where both insider and outsider adversaries may be present. Anyway, [19] envision a passive eavesdropper adversary, while the adversary anticipated in [20] does not compromise nodes originating a given message, but only the one along its path to the destination.

A more recent result addressing data availability in WSNs is [21]. It develops a scheme to maximize the amount of data recovered by the sink and shows how the proposed scheme improves data availability when a portion of the network is invalidated by natural disasters, such as a flood or an earthquake.

Benenson, et al. [22] investigated possible strategies for preventing a mobile adversary from learning certain sensed data and/or for preventing contiguous unauthorized access, once the data has been learned. Data is randomly moved around the network and an adversary who once had access to the data stored at some captured sensor, must compromise other sensors in order to retain its access to the target data. Several algorithms are introduced to provide efficient data retrieval and update.

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 [23]. The proposed forward-secure aggregate authentication techniques can efficiently provide *forward security*, i.e., having compromised a sensor, the adversary is unable to modify any data collected prior to compromise. Our focus is in this paper is quite different: we assume that the adversary is actively *pursuing* certain data and is not reluctant to delete any data it finds.

7. Conclusion

This paper represents the initial attempt to apply cryptography in the context of data survival in unattended WSNs. As we have shown, the presence of a capable focused mobile adversary, raises many challenges. However, the *good news* is that simple cryptographic defenses coupled with data mobility strategies can be of great help in ensuring data survival. We explored a number of variables and evaluated several proposed techniques. Analytical and simulation results show that proposed techniques achieve significant probabilities of data survival. Despite our simple network model, we believe that the issues raised in this paper can pave the way for further research. In our future work we plan to introduce new assumptions and variables such as communication and storage overhead, as well as new adversarial models.

References

[1] P. Pathirana, N. Bulusu, A. Savkin, S. Jha, Node localization using mobile robots in delay-tolerant sensor networks, Mobile Computing, IEEE Transactions on 4 (3) (May-June 2005) 285–296.

[2] Y. Wang, H. Wu, Delay/fault-tolerant mobile sensor network (DFT-MSN): A new paradigm for pervasive information gathering, IEEE Trans. Mob. Comput. 6 (9) (2007) 1021–1034.

[3] T. Small, Z. J. Haas, Resource and performance tradeoffs in delay-tolerant wireless networks, in: WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, ACM, New York, NY, USA, 2005, pp. 260–267.

[4] Information Processing Technology Office (IPTO) Defense Advanced Research Projects Agency (DARPA), BAA 07-46 LANdroids Broad Agency Announcement, `http://www.darpa.mil/IPTO/solicit/open/BAA-07-46_PIP.pdf` (2007).

[5] R. Ostrovsky, M. Yung, How to withstand mobile virus attacks, in: PODC, 1991, pp. 51–59.

[6] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Robust and efficient sharing of RSA functions, in: CRYPTO, 1996, pp. 157–172.

[7] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Robust threshold DSS signatures, in: EUROCRYPT, 1996, pp. 354–371.

[8] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, G. Tsudik, Catch me (if you can): Data survival in unattended sensor networks, in: IEEE PerCom, 2008, pp. 185–194.

[9] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, Proactive secret sharing or: How to cope with perpetual leakage, in: D. Coppersmith (Ed.), CRYPTO, Vol. 963 of Lecture Notes in Computer Science, Springer, 1995, pp. 339–352.

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

[11] V. Shoup, Oaep reconsidered, Cryptology ePrint Archive, Report 2000/060 (2000).

[12] M. Bellare, B. S. Yee, Forward-security in private-key cryptography, in: CT-RSA, 2003, pp. 1–18.

[13] A. Wander, N. Gura, H. Eberle, V. Gupta, S. C. Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: Percom '05, 2005, pp. 324–328.

[14] G. Ateniese, J. Camenisch, B. de Medeiros, Untraceable RFID tags via insubvertible encryption, in: CCS '05, 2005, pp. 92–101.

[15] M. Matsumoto, T. Nishimura, Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. 8 (1) (1998) 3–30.

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

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

[18] S. Chessa, P. Maestrini, Dependable and secure data storage and retrieval in mobile, wireless networks., in: DSN 2003, 2003, pp. 207–216.

[19] V. Berman, B. Mukherjee, Data security in manets using multipath routing and directional transmission, in: IEEE ICC, 2006, pp. 2322–2328.

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

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

[22] Z. Benenson, P. M. Cholewinski, F. C. Freiling, Simple evasive data storage in sensor networks, in: S. Q. Zheng (Ed.), IASTED PDCS, IASTED/ACTA Press, 2005, pp. 779–784.

[23] D. Ma, G. Tsudik, Extended abstract: Forward-secure sequential aggregate authentication, in: IEEE Symposium on Research in Security and Privacy, (S&P'07), 2007, pp. 86–91.