

Pollination: A Data Authentication Scheme for Unattended Wireless Sensor Networks

Tassos Dimitriou
Athens Information Technology
19.5 km Markopoulo Ave., 19002, Peania
Athens, Greece
Email: tdim@ait.edu.gr

Ahmad Sabouri
Chair of Mobile Business & Multilateral Security
Goethe University Frankfurt
Frankfurt, Germany
Email: ahmad.sabouri@m-chair.net

Abstract—An Unattended Wireless Sensor Network (UWSN) is a recently introduced type of sensor network in which real-time data delivery of information is replaced by periodic, offline collection of the sensed data. In this work we focus on UWSNs operating in a hostile environment where the goal of an attacker is to prevent targeted data from ever reaching the sink. More precisely, we study the Data Authentication problem in the presence of a Mobile Adversary who is aiming to target a sensor’s data and modify it without being detected.

Inspired by the pollination process in nature, we propose two schemes that diffuse data footprints in the network using message carrying “butterflies”. These schemes greatly increase the robustness of the network against data modification attempts by the mobile adversary. Extensive analytical and experimental results confirm the superiority of both *Pollination* and *Pollination Light* over previous protocols in terms of both security and communication overhead.

Keywords—Unattended Sensor Networks; Mobile Adversary; Data Authentication; Data Survival; Security and Protection

I. INTRODUCTION

Wireless sensor networks have been an attractive area of research for the last few years. Despite the hardware limitations of sensor nodes (light CPU, little memory, running on batteries, etc.) researchers have been trying to come up with solutions to problems related to routing, system architecture, middleware support, power-awareness and security. As a result, sensor networks have found many applications in various aspects of daily life. They have been used for environmental and habitat monitoring, healthcare, home automation, and traffic control. Furthermore, analysis of different operating environments and their required network models resulted in emerging new “flavors” of WSNs [1].

The concept of an Unattended Wireless Sensor Network (UWSN) is a relatively new type of sensor network ([1], [2]) where the data is *not* constantly reported to (or collected by) the sink. The data sensed by the nodes have to be accumulated for a certain amount of time because the sink is away from the network in most of the network’s lifetime. Only at the end of each collection interval, will the sink visit the sensing region and collect the data sensed by sensors.

The idea of having such a network with an itinerant sink looks realistic if we take into account the cases where the operating environment precludes the sink’s constant presence

(e.g. consider the case of an underwater network or a network sensing radioactive or nuclear material). Additional reasons for the use of UWSNs may include the large scale and wide coverage area of the network, power consumption in massive data processing and communication of data and the tendency of adversaries to target a centralized sink thus making it a single point of failure.

The unattended nature of these networks together with the data delivery procrastination and the lack of tamper resistant equipment introduces a vulnerability period where an adversary can compromise a node and manipulate or omit its data before handing it to the sink. Therefore, a mechanism is needed to ensure that the data received by the sink is *authentic*. In this paper, we aim to introduce such a data authentication mechanism where nodes collectively participate in producing fingerprints for data collected by sensors. The proposed scheme is resilient against UWSN’s mobile adversary [2] whose goal is to inject fraudulent data into the network and remain *undetected*. Such a mobile adversary is not only capable of compromising k nodes simultaneously in each round of the network collection period but can also switch to different sets of compromised nodes per round.

In our proposed scheme, we try to strengthen the network against such modification attempts by disseminating the effects of a sensor’s collected data in the key evolution process of other nodes. Thus, in order to successfully manipulate even a single data item, the adversary must have *all* (directly or indirectly) affected nodes compromised as well in order to “synchronize” them with the new value. Otherwise, the mobile sink would be able to easily detect the inconsistency in the network and realize the presence of the adversary. Our approach greatly increases the probability of data survival in the network by guaranteeing that the minimum number of nodes which will be triggered by a single data manipulation is $O(N)$, where N is the size of the network.

Organization: The rest of this paper is organized as follows. Section II gives an introduction to existing data authentication techniques in WSNs and UWSNs. In Section III, we talk about the network and adversarial model used in this work. We introduce Pollination in Section IV and provide its security analysis and cost evaluation in Section

V. Pollination’s performance is validated by the experimental results of Section VI. A suitable trade-off between security achieved and communication overhead is also proposed in Section VII. Finally, Section VIII concludes this paper.

II. RELATED WORK

In spite of the rich literature for security related research for general purpose sensor networks, there has not been much work done yet for UWSNs. The major difference which makes the existing security techniques not applicable to UWSNs is the assumption for constant presence of the sink. When we focus on the data authentication problem, UWSNs require a *bottom-up* (sensor-to-sink) mechanism to prove the authenticity of the delivered data. One of the most recent works which has provided solutions to the problem is [2]. It confirms that the prior works such as [3] and [4], where collaborative approaches have been used to detect false data along the delivery route to the sink, are not suitable for UWSNs. A similar concern exists for the proposed techniques in [5] and [6] which enhance en-route filtering provided in [4] using localization information and multipath routing, respectively.

A proper data authentication scheme for UWSNs must provide for both *forward* [7] and *backward* [8] security. The first property ensures that it will be computationally infeasible for an adversary to recover past secret keys of a compromised node if she knows the current value of the key. On the other hand, backward security guarantees that knowledge of current key cannot be used to disclose any information about future ones. A forward secure technique is resilient only against *reactive* adversaries. Such an adversary is inactive until it gets a signal that certain data must be erased. Then it wakes up and starts compromising up to k sensors per round.

A forward secure scheme for UWSNs has been proposed in [9] but cannot cope with *proactive* adversaries (such an adversary essentially starts compromising sensors at round 1, *before* receiving any information about the target sensor and the target data collection round). In a UWSN with a proactive adversary, the data authentication mechanism must necessarily be backward secure to guarantee protection of future sensor keys and data protected with these keys. However, all existing techniques which can be used to create both forward and backward secure schemes ([8], [10], [11], [12]) are not applicable in the setting of an UWSN since they assume the existence of a trusted external party.

Di Pietro *et al.* [2] recently presented a scheme for UWSNs which is resilient against both reactive and proactive adversaries. The idea is based on nodes’ collaboration in evolving cryptographic keys using Message Authentication Codes (MACs) of the previously sensed data. Although this approach provides good security guarantees, it suffers from a high transmission cost relative to the security level achieved. Our main effort in this work will be to maximize security against modification attempts while reducing communication overhead to a bare minimum.

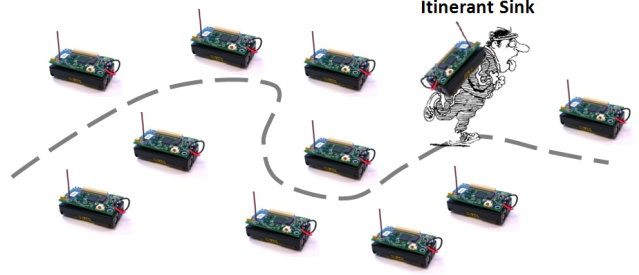


Fig. 1. A sink collecting data in a UWSN.

III. NETWORK MODEL AND ASSUMPTIONS

A. Network Model

The network setting we consider in this paper follows the one in [2] consisting of N nodes s_1, s_2, \dots, s_N uniformly distributed in the coverage area.

The network operates in units of time called *rounds* and the sensors are responsible to sense the environment once per round. The data measured by a node will be collected at every visit of the mobile sink which are at most v rounds apart (Figure 1). Therefore, the sensors are considered to have enough storage to save their own data for these v rounds. The network is also supposed to be connected at all times so that any pair of the nodes will be able to communicate. Furthermore, cryptographic capabilities are implemented in the nodes and any key establishment scheme can be utilized to bootstrap security. The sink has ample time to visit all nodes in every collection period and evaluate the trustworthiness of the collected data or re-initialize the secret keys established in the nodes, if necessary. The notation used throughout the paper is summarized in Table I.

TABLE I
NOTATION USED

ADV	the mobile adversary
N	size of the UWSN
v	max number of rounds between successive sink visits
τ	duration of a round
C_r	set of compromised sensors at round r
k	ADV 's compromise power – maximum size of any set C_r
\bar{r}	ADV 's targeted data collection round
s_i	$i \in [1..N]$, a typical sensor node
d_j^r	data sensed by s_j in round r
K_j^r	key used by s_j in round r
z_j^r	MAC of data computed by s_j at round r using K_j^r
t	number of co-authenticators used in [2]
$pollen_j^r$	the value carried by a <i>butterfly</i> initiated at node j and round r in Pollination scheme

B. Adversary Characterization

Just as the network operates in rounds, the anticipated adversary (from now on ADV) also operates in rounds. It

can compromise a set of k ($k \ll N$) nodes in each round r (denoted by C_r) and this set can have a non-empty intersection with previous compromised sets. Similar to the work in [2], we assume that the collection and compromise rounds have the same duration and they are synchronized.

Compromising a node leads to having access to all the secrets, stored information and the network traffic of that node. The goal of \mathcal{ADV} is to manipulate the target data sensed by sensor s_i at round \bar{r} ($1 \leq \bar{r} < v$) and stay *undetected*. Therefore, \mathcal{ADV} generally avoids any activity which may trigger an alarm or reveal its presence [13]. The attack will be successful if (i) the fake data passes the verification by the sink and gets accepted as the data measured by sensor s_i at round \bar{r} , and (ii), the sink does not realize the presence of \mathcal{ADV} .

\mathcal{ADV} can be a reactive adversary and start compromising its target at round $\bar{r} + 1$ after identifying it, or it can operate in proactive mode and reside in a node at round 1. In the latter case \mathcal{ADV} may be lucky and have $s_i \in C_{\bar{r}}$. As a result, it can replace the measured data before any security scheme works out. This case is not interesting from a security point of view, so we will focus on the case where $s_i \notin C_{\bar{r}}$ and \mathcal{ADV} does not have access on the node while the target data are being collected. However, \mathcal{ADV} might have the node compromised in *earlier* rounds ($r < \bar{r}$). We further assume that \mathcal{ADV} has full knowledge of the topology of the network and more importantly the defense strategy used in the UWSN.

IV. DATA AUTHENTICATION USING POLLINATION

We start with a brief introduction to the protocols in [2]. As noted in this work, simple public key signatures or Message Authentication Codes (which are referred as Basic Techniques) are not strong enough because once \mathcal{ADV} compromises a node and gets hands on its secret key, it can easily go undetected by creating a signature or MAC on the faked data.

Instead, the authors in [2] proposed the Cooperative MAC (Co-MAC) and Extensive Cooperation (ExCo) schemes in order to resolve the defects of the classical approaches. In CoMAC, each sensor j , after collecting its sensed data d_j^r at round r , constructs the corresponding authentication tag $z_j^r = MAC(K_j^r, d_j^r)$ and asks t randomly chosen peers to authenticate its round data and store the resulting tags as well. Each node utilizes the previous key and the identities of the nodes who have selected this node as co-authenticator to update their key and perform the key evolution process. To verify the authenticity of d_j^r , the mobile sink collects *all* the corresponding authentication tags and checks whether the collected tags can be generated using the claimed d_j^r . ExCo improves upon this strategy by introducing a dependency between its co-authenticators *and* the nodes which have this node as their co-authenticator using a MAC of the received tag set and its own round tag.

Pollination Scheme

The novel idea here is to propagate the effects of the current round data of each node in the network using a method

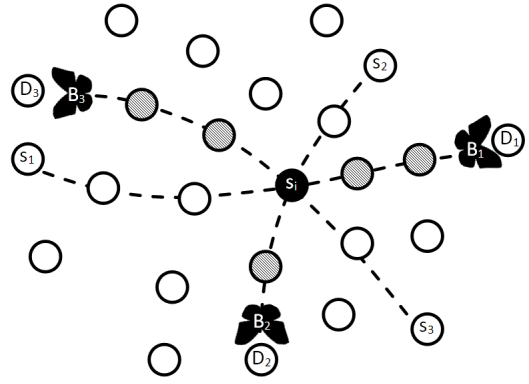


Fig. 2. Diffusion of data footprints in Pollination scheme

inspired by the pollen spreading process in nature. In our approach there are some packets traveling in the network which are playing the role of grain carrying *butterflies*. Once these butterflies pass through a node, they download their *grains* to be used by the key evolution function of that node and also pick up some *pollen* from the current round sensed data by which they “*mutate*” their carrying grains. The net effect of this procedure is that it greatly reduces the probability of the attacker modifying sensor data as it has to compromise nearly all the nodes in the network in order to stay undetected (Section V).

Figure 2 demonstrates how the butterflies spread the footprints of a sample data in the network. B_1 , B_2 and B_3 are three butterflies which have taken off from nodes S_1 , S_2 and S_3 , and are destined for nodes D_1 , D_2 and D_3 , respectively. While they are flying towards their planned destinations, they “link” visited nodes and diffuse their data footprints in the traveling path. As we have shown in the figure and explain in detail in the following paragraphs, the nodes which are visited after node s_i will experience the effects of the current round data of s_i .

We can organize the details of this approach in the following steps:

Sensing: Upon starting the round, node s_j senses the environment and measures the round value d_j^r . It uses the current key to generate a message authentication code of the data using the standard $HMAC()$ algorithm [14]. These operations are shown as Steps 1-3 in Algorithm 1.

Butterfly Take-off: Right after the sensing stage, the node initiates a message (*the Butterfly*) accompanied with the MAC of the current round data (*the Pollen*) and sends it towards a random destination (Steps 4 and 5). The sink needs to know about these destinations for the verification process but it is not necessary for the node to store them. This can be achieved easily using a pseudo-random function which is initialized by the sink. However, in order to deprive \mathcal{ADV} of discovering the future selected destinations of a compromised node, the data measured in each round as well as the messages arriving at the node can contribute in random number generation and intro-

Algorithm 1 Main Pollination procedure running at node s_j

1. Sense d_j^r
 2. Compute $z_j^r = \text{HMAC}(K_j^r, d_j^r)$
 3. Store (d_j^r, z_j^r) locally
 4. Set $\text{pollen}_j^r = z_j^r$
 5. Send pollen_j^r to a *random* destination
 6. Set $P = 0$
 - while** $\text{time} < \tau$ **do**
 7. Receive pollen_q^r
 8. Set $P = P \oplus \text{pollen}_q^r$
 9. Update $\text{pollen}_q^r = H(\text{pollen}_q^r, z_j^r)$
 10. Forward pollen_q^r
 - end while**
 11. Set $K_j^{r+1} = H(K_j^r, P)$
-

duce uncertainty in the selection of the butterfly's destination. However, as we will see in Section V-B, even without this extra countermeasure, the adversary must compromise almost the entire network before succeeding in replacing a sensed value with fake data.

Having one butterfly taking off from each node, there will be N of them flying over the network and each one will pass through \sqrt{N} hops on average. As a result, each node will have \sqrt{N} visitors on average.

Key evolution: In this step, the node waits for the butterflies to arrive (Step 7) and participate in the key evolution process. We assume the all butterflies reach their destination before the next round starts. Node s_j will use these \sqrt{N} pollen grains jointly with the current key to produce the next round secret key in Step 11. In order for the sink to be able to perform the verification process, the key evolution function is required to be insensitive to the order of the traveling messages. In other words, reordering the arrival of two messages should not change the result. A simple way to achieve this goal is to take the *XOR* result of all the received pollens and compute the hash value using the current round key. A variable P , which is initialized to zero in Step 6, in the end of the loop will be equal to

$$P = \text{pollen}_{q_1}^r \oplus \text{pollen}_{q_2}^r \oplus \dots \oplus \text{pollen}_{q_{\sqrt{N}}}^r.$$

Since each pollen is the result of an HMAC operation, P will be equal to a (pseudo) random value that can be used as input to a secure hash function $H(\cdot)$ for key evolution in Step 11. As a result, the produced key will be dependent to $O(N)$ data (this will be explained in more detail in the next section) thus greatly increasing data survivability in the network.

Using a hash function in conjunction with random data is also an effective way to provide for *forward secrecy* since an attacker after compromising the current round key will not be able to deduce any information about older keys.

Pollen mutation: Once the received pollen pollen_q^r is accounted for key evolution, the node applies function H on the received value and *updates* it using the current round data of this node (Step 9). Then it forwards the butterfly towards its

destination (Step 10). This is how a node diffuses the footprints of its data in the network. From now on, any node that this butterfly visits will carry the effect of the s_j 's data on its key evolution process.

The key evolution and pollen mutation methods effectively provide for *backward secrecy* since the attacker must have a complete picture of the network in order to affect nodes in the future. This is because the butterfly passes through $\sqrt{N}/2$ nodes on average after visiting a sampling node. So, having \sqrt{N} visitors, the number of affected nodes in the network by sample data is $O(N)$.

Verification: The sink visits every node after v rounds. This means that the sink has traversed the whole network once and obtained a full snapshot of the v previous rounds as in [2]. Knowing all the initial values for keys and the destination of each butterfly, the sink can fully simulate the behavior of the nodes and the butterflies since the routing protocol is deterministic. After calculating the keys, it can verify the authenticity of the collected data.

V. ANALYSIS AND EVALUATION

In this section, we analyze our proposed scheme regarding *storage* and *communication* costs, *security* and *robustness*, and compare it with the CoMAX and ExCo schemes from [2]. Since [2] does not provide analysis for some of these newly introduced factors, we calculate them as well. The analysis outcomes have been verified experimentally by the results that we provide in Section VI.

A. Storage and Communication Costs

Storage: Every node should be capable of storing all the data and the MAC values till the next visit of the sink. Therefore there will be at least $O(v)$ stored values for all the schemes. Pollination does not require any more information to commit to the storage so the total cost is $O(v)$. Although [2] claims that the parameter t (the number of co-authenticators) is a constant value, we need to consider it in our calculations because the survival probability is directly proportional to t (when we talk about security, we clarify how this parameter plays an important role for data survival). In both CoMAC and ExCo, a node needs to keep the list of IDs which it has received co-authentication request from, and each node receives t requests on average. Therefore the actual storage required by these schemes will be $O(vt)$.

Communication: During each round of Pollination there are N butterflies travelling in the network and each one goes through \sqrt{N} nodes on average¹. So, the number of point-to-point communications is going to be $O(N\sqrt{N})$ per round. Considering CoMAC and ExCo, every node has to send the tag (or the data) of the current round to t randomly chosen peers. As a result, N nodes send $N \cdot t$ messages through \sqrt{N} hops. Summing up these values, we have an overall

¹In evaluating communication costs we use the asymptotic cost of $O(\sqrt{N})$ for *point-to-point* routing, where N is the number of nodes in the network.

TABLE II
COMPARISON OF COSTS

	Storage	Overall Communication per round
CoMAC	$O(vt)$	$O(tN\sqrt{N})$
ExCo	$O(vt)$	$O(tN\sqrt{N})$
Pollination	$O(v)$	$O(N\sqrt{N})$

communication cost of $O(tN\sqrt{N})$. This clearly shows the dependency between security and communication overhead. For example, increasing the number of co-authenticators from $t = 2$ to $t = 4$ will double the power consumption of the network due to transmissions which is the most costly operation in a sensor node.

We have summarized the analysis results for costs' evaluation in Table II. In Section VII, we will explain how the communication cost of Pollination can be decreased even further (making it *constant* per node) without sacrificing the level of security achieved.

B. Security

Domino Effect Factor (DEF): DEF is the most important parameter which affects the probability of data survival in presence of \mathcal{ADV} . *The minimum number of nodes which will be triggered by a single data manipulation* is what we call DEF. In CoMAC, \mathcal{ADV} has to compromise $t + 1$ nodes in order to forge data. Considering the way ExCo works, this factor is going to be at most $t^2 + 2t + 1$ since a node tries to link its co-authenticators with the received co-authentication requests. Since t is constant, this gives a DEF factor equal to $O(1)$ for both CoMac and ExCO.

But in Pollination, a single modification pokes all the nodes which had the same butterflies visiting after the targeted node. As we mentioned earlier, in order to perform a successful attack \mathcal{ADV} must have $O(N)$ nodes compromised at the time of the attack. This is because a node is visited by $O(\sqrt{N})$ butterflies that travel in paths of length $O(\sqrt{N})$. Thus this requirement poses a considerable overhead to the adversary since according to our threat model the attacker can compromise at most $k \ll N$ nodes per round. This would make the attack practically infeasible since otherwise the adversary would run the risk of being detected. These findings are summarized in Table III.

TABLE III
PERFORMANCE PARAMETERS

	DEF	Communication per node per round
CoMAC	t	$O(t\sqrt{N})$
ExCo	$O(t^2)$	$O(t\sqrt{N})$
Pollination	$O(N)$	$O(\sqrt{N})$

The interesting point about Pollination is that propagation of data footprints happens in both space and time. We have already talked about the space dimension, the movement of butterflies in the network. Regarding time, consider the data footprints from node s_i which get involved in key evolution of

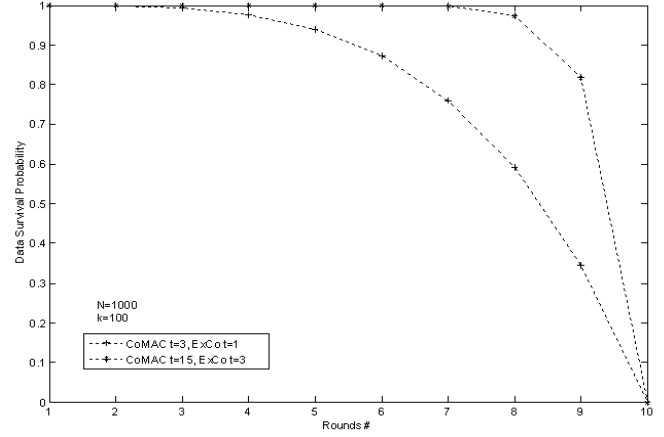


Fig. 3. Comparison between CoMAC and ExCo

s_j at round r to produce K_j^{r+1} . In the next round, butterflies who pass by s_j will have their pollens mutated as a function of z_j^{r+1} which is computed based on K_j^{r+1} . This means, these butterflies will also carry some prior effects from s_i . Therefore, as \mathcal{ADV} delays the manipulation of the target data measured at round \bar{r} , it becomes more and more difficult to cope with the domino effect introduced by our scheme (this will be further exemplified in the experimental section).

Data Survival Probability: In order for \mathcal{ADV} to have a successful attack, it has to have *all* the collaborators of the target node in its compromised set. The ideal case for \mathcal{ADV} is to use all its power and compromise k nodes around the target node and hope that none of the butterflies passing by the target node ever escapes the compromised nodes' area. This case happens if all the \sqrt{N} butterflies have one of these k nodes as their destination. For a sample butterfly, this probability is k/N because all the nodes have the same probability to be chosen as the destination. Since each butterfly operates independently, the probability of this happening for *all* \sqrt{N} butterflies is

$$P_{\text{attack}} = \left(\frac{k}{N}\right)^{\sqrt{N}} = \left(1 - \frac{N-k}{N}\right)^{\sqrt{N}} < e^{-\frac{N-k}{N}\sqrt{N}}. \quad (1)$$

So, the data survival probability which is the opposite of successful attack probability should follow a behavior like

$$P_{\text{survival}} = 1 - \left(\frac{k}{N}\right)^{\sqrt{N}}. \quad (2)$$

Another issue worth mentioning is the effect of time on the data survival probability. In both the schemes in [2] this probability *decreases* as time passes since \mathcal{ADV} gets more powerful by a factor of k (compromises k more nodes) in each round. The stored data in a node gives \mathcal{ADV} the opportunity to discover all the secret keys between two different rounds of compromising the node. Therefore, after $\lceil n/k \rceil$ rounds \mathcal{ADV} has the ability to modify any of the nodes. We have computed the attack probability in CoMAC analytically as

$$P_{\text{CoMAC}} = \frac{C(k, t+1)}{C(N, t+1)},$$

which indicates the probability of having all the $t + 1$ participants of data authentication in the compromised set. With a bit of manipulation this probability can be upper bounded as follows:

$$\begin{aligned}
 P_{CoMAC} &= \frac{C(k, t + 1)}{C(N, t + 1)} \\
 &= \prod_{i=0}^t \left(1 - \frac{N - k}{N - i}\right) \\
 &< e^{-\sum_{i=0}^t \frac{N - k}{N - i}} \\
 &< e^{-\frac{N - k}{N}(t + 1)}
 \end{aligned} \tag{3}$$

Similarly for ExCo, it can be formulated as

$$P_{ExCo} = \frac{C(k, t^2 + 2t + 1)}{C(N, t^2 + 2t + 1)} < e^{-\frac{N - k}{N}(t + 1)^2}. \tag{4}$$

Figure 3 demonstrates the comparison between CoMAC and ExCo concerning the data survival probability based on our analysis which agrees with the simulation results in [2]. It clearly shows how the probability drops when it gets closer to round $\lceil n/k \rceil$. This is not the case in Pollination because the nodes do not keep any information regarding the values participating in the key evolution process. However, even the knowledge of *all* these values does not help because \mathcal{ADV} must have $O(N)$ compromised nodes *exactly* at the attack round. For example, as we will show in the experimental section, the data survival probability for a network of 1000 nodes and an adversary with the power of compromising 100 nodes in each round will stay equal to 99% for *all* the rounds.

The robustness of Pollination with respect to attacks can also be seen by comparing expressions (1) and (4) directly. All other things being equal, the attack probability in both CoMAC and ExCo (Expressions (3) and (4), respectively) decreases exponentially as a function of t , where t must be kept *constant* so that the communication overhead is acceptable (recall Table II). However, in Pollination, the attack probability decreases exponentially as a function of \sqrt{N} .

C. Robustness

Pollination needs a deterministic routing scheme otherwise the sink will not be able to track the movement of the butterflies and hence evaluate the authenticity of the network data. Apart from the routing protocol and the underlying MAC protocol which can ensure message delivery ([15], [16]), there must be some mechanism to handle accidental node failures.

Two types of failures are foreseen here, *power failure* and *sudden crash*. The former is predictable if the node keeps track of its remaining energy, so the node can exclude itself from participating in the routing process during the next v rounds if it thinks it will not be able to operate properly until the next sink visit. This is a standard technique followed by energy aware protocols which restrict nodes participating in critical operations unless they have enough energy to cope with the requirements of the operation (see for example [17]). To handle the latter case, a simple approach would be to let the

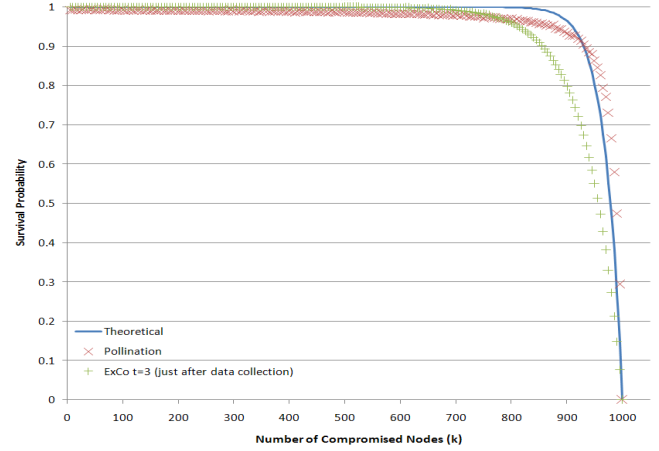


Fig. 4. Data survival probability in Pollination

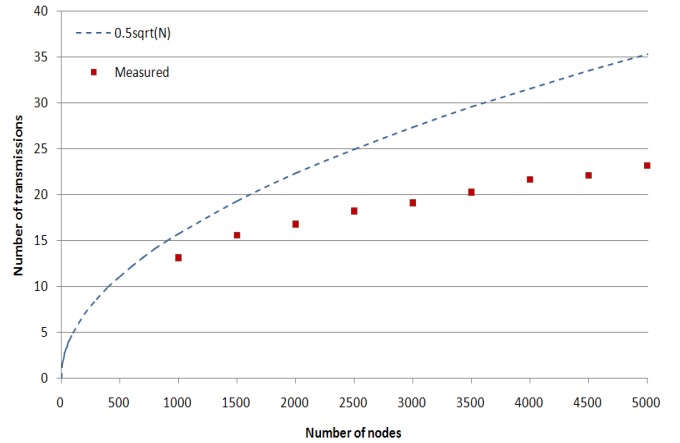


Fig. 5. Communication cost per node per round

nodes *replicate* their data in one or two neighboring nodes. In this way the sink can recover lost information in case a node suffered from a sudden crash.

VI. SIMULATION RESULTS

In this section we evaluate the effectiveness of Pollination. We generated random network topologies by placing 1000 nodes uniformly at random in the unit square with a network density (i.e. number of neighbors) equal to 8. To ensure statistical validity, each experiment was repeated 100 times. For each instance of the network, k random nodes were selected as the compromised set and one of them as the target node. This was repeated 100 times per network configuration to find out how probable it is to have a footprint of the target data outside the compromised set. We ran the simulation for $k = 1$ where only the target node is compromised to $k = N$ which assumes a very powerful \mathcal{ADV} who can compromise the entire network. Figure 4 verifies our expected behavior for the data survival probability which we computed in Section V-B (Expression 2). For comparison, the same figure also includes the behavior of ExCo for $t = 3$, before it starts to

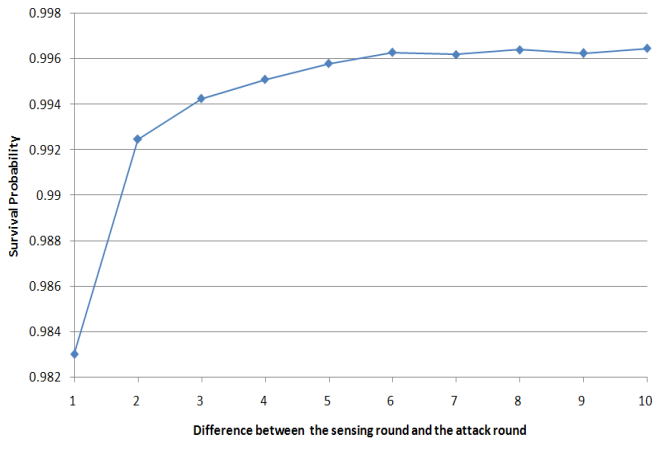


Fig. 6. Data footprint diffusion in time

drop by time as we demonstrated in Figure 3.

We have also measured the communication cost for our proposed scheme. In the analysis of Pollination we mentioned that the total number of transmissions will be equal to $O(N\sqrt{N})$ or $O(\sqrt{N})$ per node. In order to show the correctness of this analysis, we increased the network size from 1000 to 5000 and observed how many transmissions will occur in each case. Figure 5 depicts the average communication cost per node for each round as well as the $0.5\sqrt{N}$ case to ease comparison.

Another important point we want to provide by our simulation results is the propagation of data footprints in the *time* dimension. We have already mentioned that a sample data tag participates in the key evolution of other nodes and these keys will be used for the next round tag generation and consecutively pollen mutation in those nodes. So, as time passes the data footprints will diffuse more and more in the network and make it more difficult for \mathcal{ADV} to eliminate them. We have simulated this behavior and extracted the graph in Figure 6 which indicates how the probability of data survival *increases* if \mathcal{ADV} *postpones* the attack to its target. The simulation was for 100 nodes where 10 of them were compromised, and in order to get the steady state results, each experiment was repeated 500 times.

VII. TRADING SECURITY FOR SAVINGS IN COMMUNICATION

For some applications, it may be more important to have a network operating for longer periods of time than having a perfectly secured network, so having a trade-off between the security level achieved and the communication overhead might be desirable.

“*Pollination Light*” can provide this trade-off by limiting the number of butterflies flying in the network. Assuming each node initiates a butterfly with probability p , the expected number of butterflies in the network will be equal to pN . Since each one of them goes through \sqrt{N} nodes, there will be $pN\sqrt{N}$ node visits and each node will have $p\sqrt{N}$ visitors on average. As a result, the transmission cost per

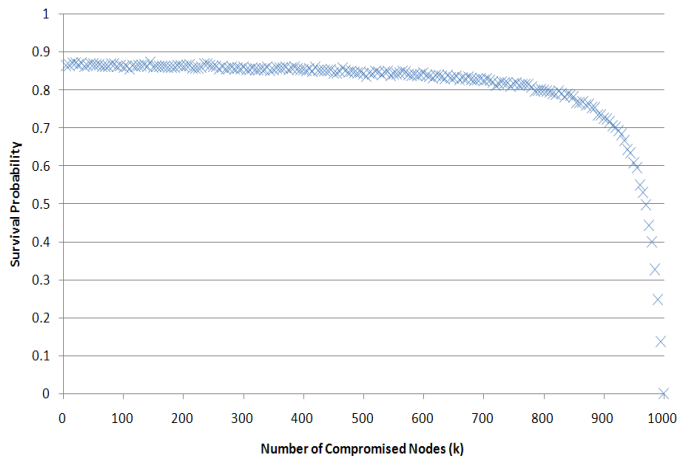


Fig. 7. Data survival probability in Pollination Light

node will be equal to $p\sqrt{N}$ as well. Now, if we want to have a *constant* number b of butterfly visits per node (and consequently constant transmission cost per node) instead of \sqrt{N} as in the original Pollination scheme, we need to set

$$p = b/\sqrt{N}, \quad (5)$$

where $b = O(1)$. Thus each node has to initiate a butterfly with probability b/\sqrt{N} (Step 5 of Algorithm 1). Notice, however, that in this case the security level drops since DEF will change to $b\sqrt{N}/2$ or in other words $O(\sqrt{N})$.

We have performed a number of simulations to investigate this trade-off. We used the same configuration as in the prior experiments ($N = 1000$), but with the additional constraint that each node initiates a butterfly with probability $p = 10/\sqrt{N}$ (i.e. $b = 10$), so that the expected number of butterflies remains constant. As it is shown in Figure 7, even with 800 compromised nodes, Pollination Light still gives a survival probability of 80% (as opposed to 96% in the original scheme). However, if we consider the footprints’ diffusion in the time dimension, Pollination Light is still going to be secure (recall Figure 6).

Simulation results also verify the analysis for DEF and the communication cost in Pollination Light. Figure 8 perfectly validates our claim about DEF in Pollination Light while Figure 9 indicates that increasing the network size does not affect the average number of transmissions per node which remains *constant* as expected.

TABLE IV
POLLINATION LIGHT PERFORMANCE PARAMETERS

	DEF	Communication per node per round
CoMAC	t	$O(t\sqrt{N})$
ExCo	$O(t^2)$	$O(t\sqrt{N})$
Pollination Light	$O(\sqrt{N})$	$O(1)$

We have summarized the performance parameters of Pollination Light in Table IV. Recall that in both CoMAC and ExCo schemes, t must be kept constant. Hence Pollination Light increases the resilience (DEF) of the network by a factor

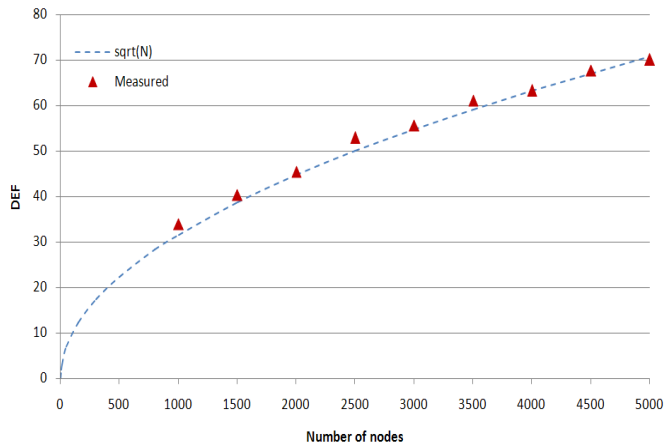


Fig. 8. DEF in Pollination Light grows as a function of \sqrt{N}

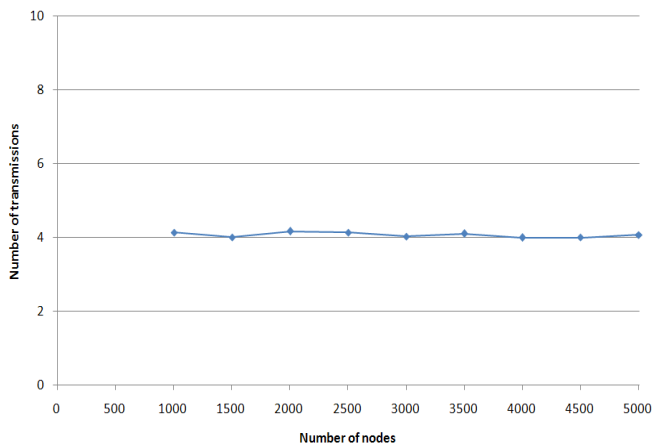


Fig. 9. Communication cost per node per round in Pollination Light

of \sqrt{N} compared to [2] while decreasing communication costs by another \sqrt{N} factor to a constant value!

VIII. CONCLUSIONS

In this work we have considered the Data Authentication problem in Unattended Wireless Sensor Networks where the delay in data collection makes nodes vulnerable to data modification attacks by a mobile adversary. Utilizing only simple cryptographic mechanisms and protocols, we have proposed two schemes, Pollination and Pollination Light, that defend against both reactive and proactive mobile adversaries (a reactive adversary starts compromising sensors after it has identified its target, while a proactive one starts compromising nodes well in advance). Pollination maximizes the level of security, forcing an adversary to compromise almost the entire network before succeeding in manipulating sensor data, while Pollination Light optimizes communication costs at a mere constant number of packets per node. Extensive analytical and experimental results confirm that both schemes outperform existing works in the area.

IX. ACKNOWLEDGEMENTS

This work has been funded by the European Community's FP7 projects LOTUS (Grant Agreement no: 217925) and SafeCity (Grant Agreement no: 285556).

REFERENCES

- [1] D. Ma, C. Soriente, and G. Tsudik, "New adversary and new threats: security in unattended sensor networks," *Network, IEEE*, vol. 23, no. 2, pp. 43–48, 2009.
- [2] R. Di Pietro, C. Soriente, A. Spognardi, and G. Tsudik, "Collaborative authentication in unattended wsns," in *Proceedings of the 2nd ACM conf. on Wireless network security*, WiSec '09, pp. 237–244.
- [3] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "Interleaved hop-by-hop authentication against false data injection attacks in sensor networks," *ACM Trans. Sen. Netw.*, August 2007.
- [4] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 839–850, 2005.
- [5] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward resilient security in wireless sensor networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '05, pp. 34–45.
- [6] Y. Zhang, J. Yang, and H. T. Vu, "The interleaved authentication for filtering false reports in multipath routing based sensor networks," in *Proc. of the 20th international conference on Parallel and distributed processing*, IPDPS'06.
- [7] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pp. 431–448.
- [8] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '02, pp. 65–82.
- [9] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, "Playing hide-and-seek with a focused mobile adversary in unattended wireless sensor networks," *Ad Hoc Netw.*, vol. 7, pp. 1463–1475, November 2009.
- [10] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong key-insulated signature schemes," in *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography*, pp. 130–144, 2003.
- [11] G. Itkis, "Intrusion-resilient signatures: generic constructions, or defeating strong adversary with minimal assumptions," in *Proceedings of the 3rd international conference on Security in communication networks*, pp. 102–118, 2003.
- [12] "Sibir: Signer-base intrusion-resilient signatures." in *CRYPTO*, 2002, pp. 499–514.
- [13] I. Krontiris, T. Giannetsos, and T. Dimitriou, "Lidea: a distributed lightweight intrusion detection architecture for sensor networks," in *Proceedings of SecureComm '08*.
- [14] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pp. 1–15.
- [15] F. Stann and J. Heidemann, "Rmst: reliable data transport in sensor networks," in *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, May 2003, pp. 102 – 112.
- [16] W. Ye, J. S. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM*, 2002.
- [17] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *Wireless Communications, IEEE Transactions on*, vol. 1, no. 4, pp. 660 – 670, Oct. 2002.