ABBA: A Balls and Bins Approach to Secure Aggregation in WSNs

Claude Castelluccia
INRIA
655, avenue de l'Europe
38334 Saint-Ismier Cedex, France
Claude.Castelluccia@inrialpes.fr

Claudio Soriente
Computer Science Department
University of California, Irvine
csorient@uci.edu

***Abstract*—Sensor networks pledge to solve many monitoring problems: thousands of small inexpensive devices can be easily deployed in any environment and can provide measurements about diverse phenomenons, such as temperature, pollution, birds migration, etc. As sensors are low-capabilities, battery powered devices, several protocols have been proposed to maximize their lifetime, but only recently research has focused on security issues such as privacy and integrity: sensors are also very easy to tamper with and usually deployed in hostile environments, where they can be easily corrupted by an attacker in order to manipulate the information provided by the network.**

In this paper, we present a novel secure data aggregation protocol that provides security and integrity for sensor networks using inexpensive cryptographic tools. Our scheme protects against both internal and external attackers and balances message size, as well as energy consumption among network nodes. It provides the sink with a great amount of information, as it is able to compute mean, standard deviation, frequency distribution, etc. of the sensed values, with only one query.

***Index Terms*—Data aggregation, Privacy, Integrity.**

I. Introduction

Sensor networks are constantly increasing in popularity in monitoring applications. Sensors are simple, inexpensive, battery powered devices with small computational power and, most of the times, wireless communication capabilities. They also have sensing capabilities to monitor light, seismic waves, vehicular traffic, etc. [1]. The general model for sensor networks [2], [3] consists of a few thousands nodes arranged in a tree topology rooted at a special node, the sink, that has no power nor computation constraints. Sensing is triggered by a query broadcasted by the sink; nodes sense a value and send their measurements through the tree back to the sink.

Because they are usually deployed in hostile environments and their replacement might be expensive or impossible, one of the first research challenges was to design sensing protocols that minimize the amount of data sent by each node and, as a consequence, maximize their battery lifetime. A very popular solution is in-network data aggregation [4]. Most of the times, the sink is not interested in single measurements but in a function of them. The main idea is to compute the function while data traverse the network, avoiding sending each measurement to the sink. As an example, if the sink is interested in the mean of the measurements, each node can add its measurement to the ones received by its children before forwarding the result to its parent; once the sum of the measurements reaches the sink, the latter divides the received value by the number of nodes in the network to get the average.

In-network data aggregation protocols are very efficient in terms of energy consumption but do not take into account security issues. Sensor are very easy to tamper with and cannot be protected by any means when they are deployed in hostile environments. If a sensor is corrupted, the attacker can arbitrarily change its sensed value. This is not different from a faulty node providing incorrect measurements or a properly working node deployed in an area where an attacker has modified the environmental conditions; i.e., lighting a match close to a sensor monitoring temperature, will sensitively increase its measurement. A very serious threat is represented by an attacker that corrupts one or more nodes, obtains their cryptographic materials, and uses them to modify network messages in order to provide the sink with false measurements. An adversary can also perform a denial of service attack blocking messages between sensors, but this kind of attack is out of the scope of this paper.

We propose a novel network aggregation protocol that mitigates the problems related to corrupted nodes in sensor networks. Our solution is based on scalar quantization and the main idea is to use a bucket with a fixed number of slots, where each slot represents an interval of the values that can be sensed in the network. Each sensor, instead of reporting its sensed value, increases by one the value of one bucket slot, similarly to voting systems where different bins represent different choices and users must state their preference inserting a ball in one of them. Surprisingly, this technique provides some interesting security features.

Our scheme has been designed with the following goals:

- *Privacy* - Node measurements should only be revealed to the sink. No external nor internal attacker should learn about other sensor measurements.
- *Robustness* - The sink must be able to detect, with high probability, malicious manipulation of the sensed values, even if a small fraction of the sensors in the network has been corrupted. The probability of detecting an attack

WIOPT 2008, 1st – 3rd Apr 2008, Berlin, Germany.
Copyright © 2011 – 2012 ICST ISBN 978-963-9799-18-9
DOI 10.4108/ICST.WIOPT2008.3183

should be proportional to the strength of the attack.

- *Low energy consumption* - Because sensors are battery powered and lifetime is a major issue, they must send a constant number of bits per query, regardless of their position in the network topology or their proximity to the sink.
- *Computation* - Because sensors are not capable of expensive computations, they should only perform simple operations such as xor, sum, multiplication, etc.
- *Versatility* - The protocol must be versatile enough to allow the sink to compute several functions of the sensed values, as mean, standard deviation, etc.

The rest of the paper is organized as follows. Section II surveys related work while section III introduces our scheme, $ABBA$. Section IV analyzes $ABBA$ computation and communication requirements, while section V analyzes its security. Section VI gives conclusion and future work.

II. Related Work

Several proposals have focused on security for low-energy sensor networks. None but [5] provides internal privacy and most of them rely on statistical instruments to detect attacks. Some proposals require much trust in the aggregators, that become very attractive targets for attackers; others are based on commit and attest paradigms that require several rounds and several messages to be sent by network nodes.

Wagner [6] defines the term *resilient aggregation* to refer to function computations that use aggregation and are robust against arbitrary changes to a subset of the sensor measurements. The authors show that some functions, like min/max computations, are inherently insecure and that a secure protocol for their computation is not likely to be found in a constrained environment such as sensor networks.

Castelluccia et al. [5] propose a variant of the one-time pad encryption scheme to provide privacy using inexpensive computations. Their scheme has very low bandwidth requirements but does not address integrity issues.

Buttyan et al. [7] introduce a scheme where data is analyzed by an aggregator, before aggregation is performed. Main drawbacks are the role of aggregators that become attractive targets for attackers and the detection algorithm based on value comparisons, that requires aggregators to have an *a priori* knowledge about values that are likely to be sensed.

Hu and Evans [8] propose a protocol based on *delayed aggregation*: instead of performing aggregation at parent nodes, it is delayed one level above; this increases bandwidth but allows detecting single corrupted nodes. Because integrity relies on intermediate nodes, those close to the sink become more appealing targets for attackers. In their scheme, corrupting two consecutive nodes in the tree topology, allows for arbitrarily changes to all values sensed in the subtree rooted at one of the two.

Yang et al. [9] introduce a scheme based on a commit and attest paradigm. In the commit phase, nodes are divided in groups and each group provides the sink with the group aggregate, while nodes commit to their measurements. The sink uses the *maximum normalized residual test* to decide which groups provided suspicious results. During the attest phase, a subset of those nodes are required to provide their measurements. Because of the outlier detection technique, the protocol is suitable only to sensor networks where all groups sense similar values. Moreover, the commit and attest paradigm requires multiple messages to detect the presence of an attacker.

Przydatek et al. [10] present another commit and attest scheme based on Merkle-Hash trees. They design several protocols for different types of queries. In their scheme much trust is required in the aggregators, that collect raw data from the sensing nodes and then engage in interactive proofs with the sink to prove the validity of committed measurements. As before, the commit and attest paradigm might involve several rounds where some nodes are required to provide their measurements without aggregation.

Sirivianos et al. [11] state the requirements for non-manipulable aggregator node election protocols. To this end, they design and compare three secure aggregator node election protocols, which randomly choose the aggregator node in a decentralized way. They use lightweight cryptography to guarantee that no party can manipulate the outcome of the election process.

III. $ABBA$

In this section we introduce $ABBA$, a novel secure data aggregation protocol for wireless sensor networks that provides privacy and integrity against external as well as internal adversaries, using very little additional bandwidth.
The main idea is to define several *bins* for different *sensing intervals* and to demand each sensor to provide its sensed value adding one *ball* in the appropriate *bin*.

Sensed values are divided in k slots, i.e., a bucket B with k slots is used. Given $[0, T[$ as the range of sensible values, slot $B[j]$ $(j = 0, ..., k-1)$ represents the number of nodes that sensed a value between $j \cdot \lceil \frac{T}{K} \rceil$ and $(j+1) \cdot \lceil \frac{T}{K} \rceil - 1$. An additional slot will be used as checksum, leading to $k+1$ slots.

TABLE I
List of used symbols

S	The sink
N	Number of nodes
T	Maximum sensible value
g	Security parameter
μ	Average of the sensed values
σ	Standard deviation of the sensed values
B	Bucket
$B[q]$	$q-th$ slot of B
U_i	$i-th$ node
K_{i_j}	U_i's $j-th$ slot key
CHK_j	$j-th$ checksum key

Tab. I lists the symbols used in the paper. Each node U_i shares $k+1$ slot keys, $K_{i_0}, ..., K_{i_k}$ with S. All nodes share k

checksum keys $CHK_0, ..., CHK_{k-1}$, also known by S. All keys are supposed to change at every query. Operations in the checksum slot are modulo g, while operation in the other slots are modulo $N \cdot T$.

For each query, a leaf node U_i sensing a value that falls in the $q-th$ slot, adds its contribution to $B[q]$, i.e., it adds 1 to the value found in that slot. Also it adds its $q-th$ checksum key to the checksum slot $B[k]$. Finally, it adds its slot keys to each of the slots of the bucket.

In detail, a node U_i senses a value, say V, and runs the following algorithm:

```
for $j = 0, ..., k-1$ do
  if $j \cdot \lceil \frac{T}{K} \rceil \leq V \leq (j+1) \cdot \lceil \frac{T}{K} \rceil - 1$ then
    $B[j] = B[j] + K_{i_j} + 1$
    $B[k] = B[k] + K_{i_k} + CHK_j$
  else
    $B[j] = B[j] + K_{i_j}$
  end if
end for
```

It then sends the bucket to its parent in the tree.

For each query, a non leaf node U_i receives a message from η of its children and sums the received buckets, slot by slot. If the node is also a sensor and not just an aggregator, it runs the above algorithm on the bucket resulting from the sum. Finally the node forwards the bucket to its parent in the tree.

Suppose that L nodes answer to a query. When S receives messages from its children, it sums the received buckets slot by slot, subtracts the slot keys of the replying nodes and checks that the resulting bucket is valid. A negative value in any slot, a number of replying node different from the expected one or an invalid checksum will cause the sink to discard the received bucket. The following algorithm is run by S on the bucket resulting from the sum of the received messages:

```
for each $U_i$ that has replied to the query do
  for $j = 0, ..., k$ do
    $B[j] = B[j] - K_{i_j}$
  end for
end for
for $j = 0, ..., k-1$ do
  if $B[j] < 0$ then
    Return: REJECT
  end if
end for
$Sum = 0$
$Nodes = 0$
for $j = 0, ..., k-1$ do
  $Sum = Sum + B[j] \cdot CHK_j$
  $Nodes = Nodes + B[j]$
end for
if $Sum == B[k]$ && $Nodes == L$ then
  Return: ACCEPT
else
  Return: REJECT
end if
```

Similar to [5], in our solution the sink needs to be aware of the replying node id's, in order to subtract the correct keys from the slots of the bucket. Because wireless sensor networks are not always reliable, it cannot be expected that all nodes reply to all requests. Therefore there needs to be a mechanism for communicating the id's of the responding nodes to the sink. The simplest approach, is for the sensors to append their respective node id to their messages [1].

In what follows, we give a small example of how $ABBA$ works. The number of nodes in the network as well as the size of the keys are somewhat unrealistic, nevertheless, we believe it will help clarifying $ABBA$ operations.

Let us assume three sensors, arranged in a network topology as shown in Fig. 1. Suppose that possible values range from 0 to 19, that the security parameter is $g = 71$ and that a bucket B with 4 slots is used. The left side of Fig. 1 shows the system parameters, the slot keys that each nodes shares with the sink, as well as the checksum keys shared among all nodes and the sink.

The status of the bucket while it traverses the network towards the sink is shown next to each node. U_1 senses a value in the range $[0-4[$ and adds 1 to the $0-th$ slot of its bucket as well as the $0-th$ checksum key to the checksum slot $(1a)$. Then it adds its slot keys to the bucket $(1b)$ and sends it towards the sink.

U_2 starts with another copy of the bucket; it senses a value in the range $[10-14[$, i.e. it adds 1 to the $2-nd$ slot as well as the $2-nd$ checksum key to the checksum slot $(2a)$. It then adds its slot keys to the bucket $(2b)$ and sends it to its parent.

U_3 adds the buckets received from U_1 and U_2 respectively $(3a)$. As it senses a value in the range $[10-14[$, it adds 1 to the $2-nd$ slot of the resulting bucket, as well as the $2-nd$ checksum key to the checksum slot $(3b)$. It then adds its slot keys $(3c)$ and sends the resulting bucket to the sink.

S receives the bucket $(4a)$ and subtracts the slot keys of U_1 $(4b)$, U_2 $(4c)$ and U_3 $(4d)$, respectively. From the resulting bucket, S learns that one node sensed a value in the range $[0-4[$ while two of them sensed a value in the range $[10-14[$. To check the integrity of the bucket, S makes sure that $B^k = \sum_{j=0}^{j=k-1} B^j CHK_j$.

IV. Performance Analysis

As the sink is provided with all measured values, the protocol is suitable for different computations, such as mean, median, mode, frequency distribution, etc. Nevertheless, the accuracy of the results is indirectly proportional to the width

[1]Depending on the number of nodes that respond to a query, it might be more efficient to communicate the id's of non-responding sensors.

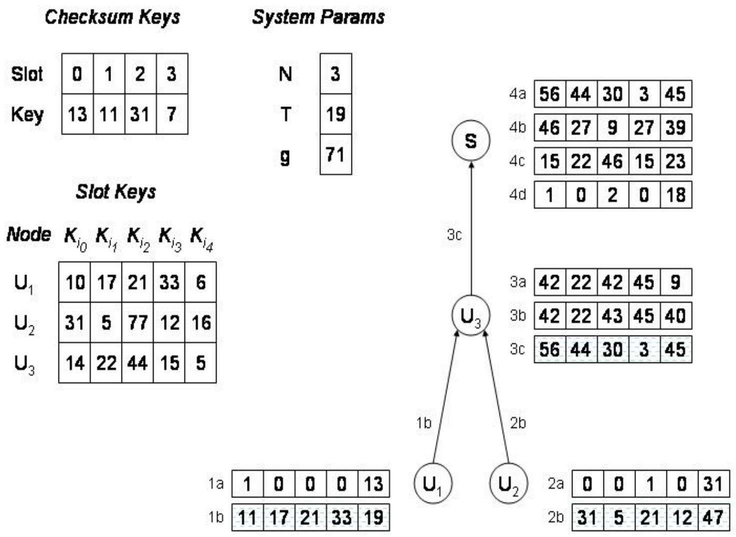

Fig. 1. Example of $ABBA$ in a network with three nodes.

TABLE II
MESSAGE SIZE OF *NO-AGG* AND $ABBA$.

	NO-AGG						ABBA			
	1024 nodes			2048 nodes			1024 nodes		2048 nodes	
	Degree			Degree			Slots		Slots	
Depth	2	4	8	2	4	8	10	20	10	20
1	44457	29667	50895	89001	118775	50895	180	280	190	300
2	22185	7395	6351	44457	29667	6351				
3	11049	1827	783	22185	7395	783				
4	5481	435	87	11049	1827	87				
5	2697	87		5481	435					
6	1305			2697	87					
7	609			1305						
8	261			609						
9	87			261						
10				87						
$Mean$	697	407	299	1567	927	647	180	280	190	300
$Std\ Dev$	2669	1803	1751	3917	3209	2103	0	0	0	0

of the slots. In some applications, different width slots might be used to gain accuracy. With fixed width slots, the size of the bucket, as well as the number of bits sent by each node, is $k \cdot log(N \cdot T) + g$. Because the only scheme that provides the sink with the same amount of information is the one where no in-network aggregation (*NO-AGG* in the rest of the paper) is performed, we compare message sizes of $ABBA$ with message sizes of protocols where raw data is forwarded from each node to the sink.

Tab. II compares $ABBA$ and *NO-AGG*, showing the number of bits sent by a node, based on its depth, in a tree with 1024 or 2048 nodes. For *NO-AGG*, we consider a 7 bit value and trees with degree 2, 4 and 8, respectively. For $ABBA$, we consider $[0 - 128[$ as the range of sensible values and 10 or 20 slot buckets. For both protocols we consider an 80 bit checksum.

An interesting property of $ABBA$ is that all nodes send the same amount of data, independently of their distance (number of hops) to the sink; in *NO-AGG*, nodes close to the sink must forward many messages that will quickly deplete their batteries. Thus, $ABBA$ helps balancing energy consumption and simplifies cluster-head election issues. Even though our analysis focus on tree structured networks, $ABBA$ works for any network topology where each node is aware of its next hop to the sink.

Nodes perform $\eta \cdot (k+1)$ modular additions where η is the number of children that replied to the query (if the node has no children, it performs $k+1$ additions).

For each query, each sensor must use $k+1$ fresh slot keys as well as k fresh checksum keys. Slot keys, that are shared only with the sink, might be produced from a unique seed, known to the sink, and a common pseudo-random function:

for each query, the function is iterated $k+1$ times and the results are used as slot keys. Checksum keys can be produced in a similar fashion but, as they have to be shared by all nodes, a common seed must be used. Such key computation also provides forward security, i.e., an adversary corrupting a node, will not be able to compute the keys that node has used before being corrupted and will not be able to learn the values sensed during previous queries.

V. Security Analysis

Slot keys guarantee privacy of sensed value only if buckets sent by leaf nodes are initialized with random values. Otherwise, an ubiquitous eavesdropper listening to messages sent by leaf nodes will learn their keys with high probability. If a leaf node U_l senses a value in the $j-th$ slot, its outgoing bucket will store its slot keys in slot $B[0], ..., B[j-1], B[j+1], ...B[k-1]$, while slot $B[j]$ will store $K_{l_j}+1$. Listening to all ingoing and outgoing messages, an eavesdropper can learn the slot keys of all nodes and use them to break the privacy of the system, i.e. to learn sensed values. Key leakage can be easily prevented by initializing the bucket slots at leaf nodes, with random values known to the sink, in a similar fashion to which keys are computed.

The most dangerous threat for sensor networks consists of an attacker that corrupts one or more sensors and uses their cryptographic material to provide the sink with false measurements as if they were sent by honest sensors.

Consider a sensor network for fire detection where the sink believes there is a fire if most nodes report a temperature above a threshold value. If the attacker wants the sink to believe there is a fire, he cannot just provide a few values above the threshold, as the sink would regard those measurements as sent by defective nodes. Indeed, the attacker should shift the mean of the distribution from its real value, to a target value above the threshold. If sensed values are reported using a bucket, as in $ABBA$, the attacker goal will be to move an appreciable number of measurements from the slots around the real mean to the slots corresponding to his target mean.

Analyzing the security of $ABBA$, we consider an adversary that is aware of the mean and the standard deviation of the sensed values. Despite such a strong assumption, $ABBA$ has high probability of discovering malicious manipulation of the sensed data. If the attacker has to estimate mean and standard deviation of the sensed values, perhaps using the measurement of its corrupted nodes, probability of being discovered is even higher, i.e., $ABBA$ is even more secure.

We consider an external as well as an internal adversary. Both of them are able to intercept, modify and reinject all packets that travel in the network. An external attacker does not control any nodes, so he can perform a denial of service attack intercepting packets and modifying the values of the bucket so that the checksum test at the sink fails. It is easy to see that the probability of changing any value in the bucket and resulting in a valid checksum is inversely proportional to the size of the modulo used in the checksum slot (g).

An internal attacker controls a corrupted node and knows its slot keys as well as the checksum keys. As the latter are shared among all nodes, corrupting more than one node will not improve the attacker capabilities. We suppose that he knows the expected distribution of the results, its mean and its standard deviation. Corrupting a node, he might be able to modify the results, moving contributions from one slot to another. In order to move w contributions from slot $B[i]$ to slot $B[j]$ the attacker must do the following:

1) Subtract w from $B[i]$
2) Subtract $w \cdot CHK_i$ from $B[k]$
3) Add w to $B[j]$
4) Add $w \cdot CHK_j$ to $B[k]$

However, decreasing the value of a slot, say $B[i]$, might lead to a negative value that will reveal to S the malicious manipulation of the bucket. Note that in order to discover the attack, $B[i]$ must be decremented by a value greater than the number of nodes that sense a value between $i \cdot \lceil \frac{T}{K} \rceil$ and $(i+1) \cdot \lceil \frac{T}{K} \rceil$. That is, even though a slot reaches a negative value because of the attacker, other nodes adding their contributions on the way through S, might increase that slot value above 0 and conceal the attack to the sink.

Suppose that an attacker has corrupted a node U_q. Suppose he wants to shift the mean from μ to μ_{att} and/or he wants to change the standard deviation from σ to σ_{att}. Knowing μ and σ he might estimate the distribution and modify the bucket, slot by slot, based on the differences between his estimate distribution and the target one. Let B_{est} be the estimated bucket and B_{att} the target one, the attacker runs the following algorithm:

```
**for** $i = 0, ..., k$ **do**
  **if** ${B_{est}}^i > {B_{att}}^i$ **then**
    $B^i = B^i - ({B_{est}}^i - {B_{att}}^i)$
    $B^k = B^k - ({B_{est}}^i - {B_{att}}^i)CHK_i$
  **else**
    $B^i = B^i + ({B_{att}}^i - {B_{est}}^i)$
    $B^k = B^k + ({B_{att}}^i - {B_{est}}^i)CHK_i$
  **end if**
**end for**
```

From the above algorithm, it is clear that the more B_{att} differs from B_{est}, the lower will be the attacker probability of success. Since the attacker knows σ and μ, the estimate distribution B_{est} will be a good approximation of the real one. If we relax the latter hypothesis, the attacker probability of success will sensibly decrease.

If the attacker tries to shift the mean of the distribution of just a few slots, he will succeed and remain undetected with high probability. Nevertheless, such an attack will not change much the information derived from the query by the sink, i.e., the sink will not trigger an alarm. To trigger the alarm, the attacker must be aggressive and significantly move the distribution of the sensed value. As shown in our simulation results, $ABBA$ detects such attacks with high probability.

Fig. 2 shows the attacker success probability when he tries to modify the distribution of the sensed values in a network with 1024 or 2048 nodes. A 10 slot bucket is used and the original mean falls in the $5-th$ slot. As shown, the attacker success probability decreases as the slot of the target mean moves away from the slot of the original one. For example, probability of moving the mean to the $8-th$ slot remaining undetected is about 0.1. It is also shown that the number of nodes in the network slightly affect the attacker success probability.

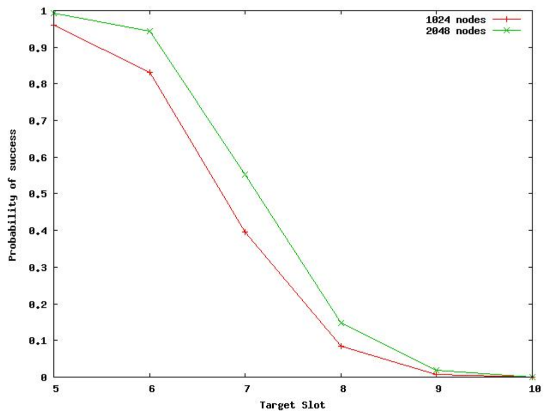

Fig. 2. Attacker Success Probability

Fig. 3 is similar to Fig. 2 but it considers buckets with different number of slots. In the 10-slot bucket, the mean falls in the $5-th$ slot and the curve shows the probability of shifting it up to the $10-th$ one, without being discovered by the sink. In the 20-slot bucket, the mean falls in the $10-th$ slot and the curve shows the probability of shifting it up to the $20-th$ one.

Let B_{10} be a 10-slot bucket and let R_{10} be the width of each slot. Let B_{20} be a 20-slot bucket and let R_{20} be the width of its slot. Given the same range of sensible values, $R_{10} = 2 \cdot R_{20}$. To increase the mean from μ to $\mu + k$ in B_{10}, the attacker has to shift the mean of $\frac{k}{R_{10}}$ slots. To achieve the same result in B_{20}, the attacker has to shift the mean of $\frac{k}{R_{20}} = 2 \cdot \frac{k}{R_{10}}$ slots. As shown if Fig. 3, doubling the distance between the original mean slot and the target one, dramatically reduces the attacker success probability.

1) Analytical Analysis: In what follows, we provide analytical analysis of $ABBA$ to validate our simulation results. We derive the success probability of an internal attacker, i.e., his probability of modifying the distribution of sensed data, without being discovered, that is, without any slot with negative values. Assume that:

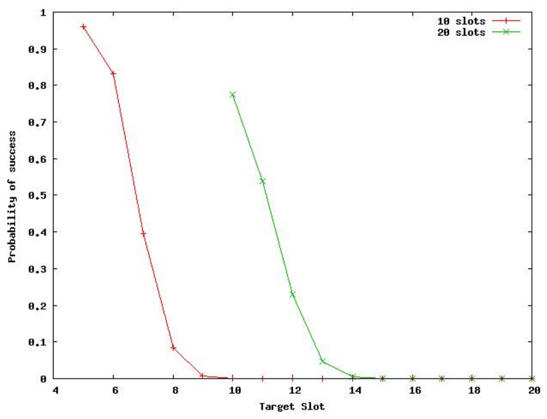

Fig. 3. Attacker Success Probability (2)

- Sensed data follow a given distribution whose probability density function (pdf) is defined by $f(u; \mu, \sigma)$, where μ is the mean and σ the standard deviation,
- The bucket has k slots.
- Sensible data range is $[0; kR]$, where $R = \lceil \frac{T}{K} \rceil$, i.e., the width of a slot.
- The network is composed of n sensors.

Given that slot B^i covers the range $[x_i, x_{i+1}]$, where $x_i = iR$ and $x_{i+1} = (i+1)R$, the probability p_i that a sensed value fall in that slot is defined by:

$$p_i = \int_{x_i}^{x_{i+1}} f(x, \mu, \sigma) dx \tag{1}$$

We are now interested in computing the distribution of the values of slot B^i, for any i. We therefore have a "bins and balls" problem, with n balls that fall in k bins according to a given distribution.

It is known that the number of balls that fall in slot B^i follows a binomial distribution. In other words, the probability that B^i contains j balls, with $0 \leq j \leq n$ is defined by:

$$P_i[j] = C_n^j \cdot p_i^k \cdot (1 - p_i)^j \tag{2}$$

This distribution can be approximated by a gaussian distribution of mean $n \cdot p_i$ and variance $n \cdot p_i \cdot (1 - p_i)$.

To indicate that a random variable X is normally distributed with mean μ and variance σ^2, we write:

$$X \sim N(\mu, \sigma^2) \tag{3}$$

Therefore, the distribution of number of balls B_i in B^i follows the gaussian distribution.

$$B_i \sim N(n \cdot p_i, n \cdot p_i \cdot (1 - p_i)), \tag{4}$$

where p_i is defined by equation 1.

The attacker transforms the bucket by subtracting from it n values that follow a gaussian distribution $N(n \cdot p_i, n \cdot p_i \cdot (1-p_i))$. He also adds n values that follow the target gaussian distribution $N(n \cdot pt_i, n \cdot p_i \cdot (1-pt_i))$, where $pt_i = \int_{x_i}^{x_{i+1}} ft(x, \mu', \sigma')dx$ and $ft(x, \mu', \sigma')$ is the target distribution of the sensed data.

It is known that if $X \sim N(\mu_x, \sigma_x^2)$ and $Y \sim N(\mu_y, \sigma_y^2)$ are independent normal random variables then the sum is normally distributed with

$$U = X + Y \sim N(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2). \tag{5}$$

Therefore, after manipulation, the distribution of the number of balls B_i' in a given slot B^i is defined as follows:

$$B_i' \sim N(n \cdot pt_i, 2 \cdot n \cdot p_i \cdot (1-p_i) - n \cdot pt_i \cdot (1-pt_i)), \tag{6}$$

where $pt_i = \int_{x_i}^{x_{i+1}} ft(x, \mu', \sigma')dx$.

We are now interested in computing the probability that the B_i' is smaller than 0, i.e. $F(0; n \cdot pt_i, sqrt(2 \cdot n \cdot p_i \cdot (1-p_i) - n \cdot pt_i \cdot (1-pt_i)))$, where $F(x; \mu, \sigma)$ is the cumulative distribution function (cdf).

The cumulative distribution function of a normal distribution $N(\mu, \sigma^2)$ is expressed as follows:

$$F(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) du \tag{7}$$

$$= \Phi\left(\frac{x-\mu}{\sigma}\right), \tag{8}$$

or

$$F(x;\mu,\sigma) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]. \tag{9}$$

An attack is successful (i.e. undetected by our algorithm) if no slot reaches a value smaller than 0. Therefore the success probability is defined by:

$$P_{success} = \prod_{i=1}^{k} (1 - F(0; \mu_i', \sigma_i')). \tag{10}$$

where
$\mu_i' = n \cdot pt_i$ and $\sigma_i' = sqrt(2 \cdot n \cdot p_i \cdot (1-p_i) - n \cdot pt_i \cdot (1-pt_i))$.

If we assume the sensed data follows also a gaussian distribution $N(\mu_s, \sigma_s^2)$ and the target distribution is also gaussian and defined as $N(\mu_{st}, \sigma_{st}^2)$, then

$$p_i = F(x_{i+1}; \mu_s, \sigma_s) - F(x_i; \mu_s, \sigma_s) \tag{11}$$

and

$$pt_i = F(x_{i+1}; \mu_{ts}, \sigma_{ts}) - F(x_i; \mu_{ts}, \sigma_{ts}) \tag{12}$$

, where $F(u)$ is defined by equation 9.

Note that our analysis is general enough to consider different types of distributions. In some applications, the sensed value will follow a uniform distribution in $[0; kR]$ in normal situations and will move to a gaussian distribution $N(\mu_{alarm}, \sigma_{alarm}^2)$ if something wrong happens (like a fire) and an alarm must be set.

The goals of an attacker can then be:

- To generate an alarm by moving the uniform distribution to a gaussian one. In this case equation 10, with $p_i = 1/k$ and $pt_i = F(x_{i+1}; \mu_{alarm}, \sigma_{alarm}) - F(x_i; \mu_{alarm}, \sigma_{alarm})$, can be used to compute the success probability of the attacker.
- To hide an abnormal situation by moving the gaussian distribution back to a uniform one. In this case equation 10, with $p_i = F(x_{i+1}; \mu_{alarm}, \sigma_{alarm}) - F(x_i; \mu_{alarm}, \sigma_{alarm})$ and $pt_i = 1/k$, can be used to compute the success probability of the attacker.

VI. Conclusion

In this paper, we introduced $ABBA$, a new in-network data aggregation protocol for sensor networks, that provides privacy and security against both external and internal attackers. $ABBA$ requires each sensor to perform inexpensive computations and to send a fixed amount of data, regardless of its position in the network. Future work includes implementation of $ABBA$ using nesC for TinyOS and real testing on MICA Motes.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, *A survey on sensor networks*, IEEE Communications Magazine, 2002.

[2] G. J. Pottie and W. J. Kaiser, *Wireless Integrated Network Sensors*, Communications of the ACM, 2000.

[3] C. Shen, C. Srisathapornphat and Ch. Jaikaeo, *Sensor Information Networking Architecture and Applications*, IEEE Personel Communication Magazine, 2001.

[4] C. Intanagonwiwat, D. Estrin, R. Govindan and J. Heidemann, *Impact of network density on data aggregation in wireless sensor networks*, Technical Report, University of Southern California, 2001.

[5] C. Castelluccia, E. Mykletun and G. Tsudik, *Efficient aggregation of encrypted data in wireless sensor networks*, 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05), 2005.

[6] D. Wagner, *Resilient aggregation in sensor networks*, ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04), 2004.

[7] L. Buttyan, P. Schaffer and I. Vajda, *Aggregation with Attack Detection in Sensor Networks*, 4th IEEE International Conference on Pervasive Computing and Communications, 2006.

[8] L. Hu and D. Evans,*Secure Aggregation for Wireless Networks*, Symposium on Applications and the Internet Workshops (SAINT'03), 2003.

[9] Y. Yang, X. Wang, S. Zhu and G. Cao, *SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks*, 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'06), 2006.

[10] B. Przydatek, D. Song and A. Perrig, *SIA: Secure information aggregation in sensor networks*, 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys'03), 2003.

[11] M. Sirivianos, D. Westhoff, F. Armknecht, J. Girao, *Non-manipulable Aggregator Node Election Protocols for Wireless Sensor Networks*, Proceedings of the International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'07), 2007.