IJRRECS/October 2013/Volume-1/Issue-6/1542-1553 **ISSN 2321-5461**

INTERNATIONAL JOURNAL OF REVIEWS ON RECENT ELECTRONICS AND COMPUTER SCIENCE

ACHIEVING DATA SECURITY IN WIRELESS NETWORK BY DATA AGGREGATION

Vikram Reddy Burri[1], Dr.S.Kalaimagal[2]

[1]M.Tech (SE), Dept of CSE, Padmasri Dr. B.V.Raju Institute of Technology, Vishnupur,Narsapur, Medak, A.P, India
Email Id: vikramreddy1297@gmail.com

[2]Professor, Dept of IT, Padmasri Dr. B.V.Raju Institute of Technology, Vishnupur, Narsapur, Medak, A.P, India
Email Id: kalaimagal.s@bvrit.ac.in

ABSTRACT:

Wireless sensor networks are envisioned to be economic solutions to many important applications, such as real-time traffic monitoring, military surveillance, and homeland security. A sensor network may consist of hundreds or thousands of low-cost sensors, each of which acts as an information source, sensing and collecting data from the environment for a given task.In a large sensor network, in-network data aggregation significantly reduces the amount of communication and energy consumption. Recently, the research community has proposed a robust aggregation framework called synopsis diffusion which combines multipath routing schemes with duplicate-insensitive algorithms to accurately compute aggregates e.g., predicates Count, Sum in spite of message losses resulting from node and transmission failures. However, this aggregation framework does not address the problem of false sub-aggregate values contributed by compromised nodes resulting in large errors in the aggregate computed at the base station, which is the root node in the aggregation hierarchy. This is an important problem since sensor networks are highly vulnerable to node compromises due to the unattended nature of sensor nodes and the lack of tamper-resistant hardware.

Keywords: wireless sensor networks, aggregation, synopsis diffusion, tamper-resistant hardware.

1. INTRODUCTION:

Wireless sensor networks (WSNs) are increasingly used in several applications, such as wild habitat monitoring, forest fire detection, and military surveillance. After being deployed in the field of interest, sensor nodes organize themselves into a multi hop network with the base station as the central point of control. Typically, a sensor node is severely constrained in terms of computation capability andenergy reserves. A straightforward method to collect the sensed information from the network is to allow each sensor node's reading to be forwarded to the base station, possibly via other intermediate nodes, before the base station processes the received data.

However, this method is prohibitively expensive in terms of communication overhead or energy spent. In large WSNs, computing aggregates in-network i.e., combining partial Results at intermediate nodes during message routing significantly reduces the amount of communication and hence the energy consumed. An approach used by several data acquisition systems for WSNs is to construct a spanning tree rooted at the base station, and then perform in-network aggregation along the tree. The important aggregates considered by the research community include Count, and Sum. Note that it is straightforward to generalize these aggregates to predicate Count e.g., number of sensors whose reading is higher than 100 unit and Sum. Furthermore, Average can be computed from Count and Sum. A Sum algorithm can be also extended to compute Standard Deviation and Statistical Moment of any order.

Tree-based aggregation approaches are not resilient to communication losses resulting from node and transmission failures, which are relatively common in WSNs. To address this problem, the research community has proposed the use of multipath routing techniques for forwarding sub aggregates. For aggregates such as Min and Max, which are duplicate-insensitive, this approach provides a fault-tolerant solution. However, for duplicate-sensitive aggregates, such as Count and Sum, multipath routing leads to double-counting of sensor readings. Recently, several researchers have presented clever algorithms to solve the double-counting problem associated with multipath approaches. A robust and scalable aggregation framework called synopsis diffusion has been proposed for computing duplicate-sensitive aggregates, such as Count and Sum. This approach uses a ring topology where a node may have multiple parents in the aggregation hierarchy, and each sensed value or sub aggregate is represented by a duplicate-insensitive bitmap called synopsis.

However, most of the existing in-network data aggregation algorithms have no provisions for security. A compromised node might attempt to thwart the aggregation process by launching several attacks, such as eavesdropping, jamming, message dropping, message fabrication, and so on.

Existing System:

The Tiny Aggregation Service (TAG) to compute aggregates, such as Count and Sum, using tree-based aggregation algorithms were proposed. Moreover, tree-based aggregation algorithms to compute an order-statistic have been proposed. To address the communication loss problem in tree-based algorithms the authors designed an aggregation framework called synopsis diffusion to compute Count and Sum, which uses a ring topology. These works use duplicate-insensitive algorithms for computing aggregates based on the algorithm for counting distinct elements in a multi set. Several secure aggregation algorithms have been proposed assuming that the base station is the only aggregator node in the network. It is not straightforward to extend these works for verifying in-network aggregation unless we direct each node to send an authentication message to the base station, which is a very expensive solution. Only recently, the research community has been paying attention to the security issues of hierarchical aggregation.

A tree-based verification algorithm was designed by which the base station can detect if the

final aggregate, Count or Sum, is falsified. We are unable to extend this idea for verifying a synopsis because the synopsis computation is duplicate-insensitive. A verification algorithm for computing Count and Sum within the synopsis diffusion approach was designed. Our algorithm has some similarity with except the fact that our algorithm attempts to further reduce the communication overhead in a novel approach. In addition, we provide extensive theoretical analysis to find the best tradeoff between the security and communication overhead. Recently, a few novel protocols have been proposed for secure out sourced aggregation; however, these algorithms are not designed for WSNs.

Proposed System:

In this project, we design an algorithm to compute aggregates, such as Count and Sum, and to enable the base station to verify if the computed aggregate is valid. We call this algorithm the verification algorithm, though strictly speaking, it is an aggregate computation and verification algorithm. The key observation whichwe exploit to minimize the communication overhead of this algorithm is that to verify the correctness of the final synopsis (the aggregate of the whole network) the base station does not need to receive authentication messages from all of the nodes. We validate the performance of our algorithm via both theoretical analysis and simulation. It is to be noted that while our algorithm is designed having WSNs in mind, it is straightforward to extend our solution for secure aggregation query processing in a large-scale distributed database system over the Internet.

Literature Survey

Wireless sensor networks are envisioned to be economic solutions to many important applications, such as real-time traffic monitoring, military surveillance, and homeland security. A sensor network may consist of hundreds or thousands of low-cost sensors, each of which acts as an information source, sensing and collecting data from the environment for a given task. In addition, there may also exist one or more base stations (or data sinks) which subscribe to specific data streams by distributing interests or queries. The sensors in the network then push relevant data to a querying base station (BS). However, it is very inefficient for every sensor node to report their raw data back because every data packet need traverse many hops to reach the BS and especially sensor nodes are often constrained by scarce resources in memory, computation, communication, and battery. On the other hand, as in many cases sensor nodes in an area detect the common phenomena; there is high redundancy in their raw data. Thus, reporting raw data is often unnecessary.

Recently many data aggregation protocols have been proposed to eliminate the data redundancy in sensor data of the network, hence reducing the communication cost and energy expenditure in data collection. During a typical data aggregation process, sensor nodes are organized into a tree hierarchy rooted at a BS. The non-leaf nodes act as aggregators, fusing the data collected from their child nodes before forwarding the results towards the BS. In this way, data are processed and fused at each hop on the way to the BS, and communication overhead can be largely reduced.Hop-by-hop aggregation, however, opens a new door to false data injection attacks. Sensor nodes are often deployed in open and

unattended environments, so they are vulnerable to physical tampering due to the low manufacturing cost. An adversary can obtain the confidential information (e.g., cryptographic keys) from a compromised sensor and reprogram it with malicious code. The compromised node may then report an arbitrary false fusion result to its parent node in the tree topology, causing the final aggregation result to far deviate from the true measurement. This attack can become more damaging when multiple compromised nodes collude in injecting false data.

The above attack is extremely difficult, if not impossible, to prevent or detect. From the viewpoint of information theory, data aggregation is a lossy data compression process because all the individual sensor readings are lost in the per-hop aggregation process. Hence, it is impossible for the BS to verify the correctness of an aggregated result without knowing the original readings. Unfortunately, the requirement of knowing the original readings effectively preclude any data aggregation techniques.

We assume a sensor network consisting of a large number of resource-limited sensor nodes. In addition, there exists a powerful BS that connects the sensor network to the outside infrastructure such as the Internet. As in other data aggregation protocols, we assume a topological tree rooted at the BS. There are various methods for constructing the aggregation tree according to different application requirements. However, SDAP does not rely on a specific tree construction algorithm as long as there is one. To concentrate on the security aspects of data aggregation, we will not address the general issues regarding data aggregation, e.g., what sensor applications might benefit from the technique of data aggregation or how to ensure time synchronization among nodes.

In a real application, a topology tree may be dynamic due to node or link failures. In Tiny OS, a beaconing message is flooded every 30 seconds to reconstruct the broadcast tree. Clearly, it will be too costly for the BS to keep track of the network topology for every topology change, because every topology discovery may require every node to report its parent/child information to the BS. As such, in our scheme, we assume that the BS does not know the shape of the tree and its distance (in number of hops) from every node although it may want to discover the tree topology occasionally for other purposes. We also assume there is a reliable transmission mechanism, for example, by using a link-layer hop-by-hop acknowledgment protocol. Thus, the various types of packets in our scheme will not be lost.

Key Setup: We assume the BS cannot be compromised and it has a secure mechanism to authenticate its broadcast messages to all the nodes in the tree and every node can verify the received broadcast messages. We also assume every sensor node has an individual secret key shared with the BS. Further, there is a unique pair wise key shared between each pair of neighboring nodes.

Attack Model

Since a standard authentication primitive , e.g., message authentication code (MAC)s, can be employed to easily defeat an outsider adversary (who do not have any authentication keys) from launching many attacks, we assume an adversary can compromise a (small) fraction of sensor nodes to obtain the keys as well as reprogram these sensor

with attacking code. There may be multiple potential attacks against a tree-based aggregation protocol. One type of attacks is behavior-based, in which the goal of an attacker is to disrupt the normal operation of the sensor network. For example, once a sensor node in the tree is compromised, it can attack the underlying routing protocol; drop other nodes' readings on purpose, or cause denial of message attacks to deprive other nodes from receiving broadcast messages of the BS.

In this paper, however, we are not addressing any of these behavior based attacks; instead, we focus on defending against *false data injection attacks* where the goal of an attacker is to make the BS to accept false sensor reports. In many situations, values received by the BS provide a basis for critical decisions; hence, false or biased values may cause catastrophic consequences. For example, when forwarding other sensor nodes' reported values, a compromised node may modify their values; it may also forge some false sensor readings on its own behalf. Becausethe measurements of the physical world are inherently noisy, if an attacker forges sensor readings that have negligible influence on the final aggregation result, he gains little. Therefore, we assume that an attacker aims to inject false values that deviate from the true measures in a noticeable scale. Apparently, the attacker does not want to be detected when launching this attack.

In particular, in the context of data aggregation, an aggregate usually contains not only a data value computed for the required aggregation function but also a count value indicating the number of sensor nodes involved in the aggregation operation. Clearly, an attacker can forge an unusual false data value as well as a large count value to make its false data account for a large portion of the final aggregated result. We refer to these two types of attacks *value changing attack* and *count changing attack*, respectively.

Wireless sensor networks will consist of large numbers of small, battery-powered, wireless sensors. Deployed in an adhoc fashion, those sensors will coordinate to monitor physical environments at fine temporal and spatial scales. Wireless sensor networks will be autonomously deployed in large numbers. Energy-efficiency is a key design criterion for these sensor networks.

A monitoring infrastructure will be a crucial component of a deployed sensor network. Such an infrastructure indicates node failures, resource depletion, and other abnormalities. Our first contribution is architecture for sensor network monitoring infrastructures, one that consists of three classes of software. The first class of software *continuously* collects aggregates of network properties (we call them network *digests*) in the background. Triggered by sudden changes in these properties, *scans* can be invoked to provide global, yet aggregated, views of system state. Such views can indicate the location of performance problems or impending failure within the network. *Dumps* can then be used to collect detailed node state to debug the problem. These three pieces of software are invoked at different spatial and temporal scales, and will allow accurate, yet low-overhead sensor network monitoring. These observations lead to two key constraints in the design of protocols for digest computation. First, digest protocols must be *aggressively* energy-efficient, far more so than other

components of the system. Second, because there isn't a natural initiator for a digest (*e.g.*, a user node) the routing structures for digest computations must be autonomously derived. To achieve aggressive energy-efficiency, we propose to piggyback digest computation messages on neighbor-to-neighbor communication. We observe that many proposed sensor network protocols for medium access and for topology control include periodic beaconing. Digests, being small by definition, can easily be piggybacked on such communication.

While not it a new idea, this approach *seems almost necessary* to achieve very low energy expenditures for digest computation. This approach trades-off latency for energy savings. We quantify this trade-off in a later section. We observe that some decomposable digest functions like *min* and *max* can be computed using a technique we call *digest diffusion*1. For example, suppose we are interested in computing a digest that represents the value of minimum energy at any node in the network; call this value Emin. Each node periodically broadcasts to its neighbors (*e.g.* by piggybacking on other messages) its own energy, as well as its current estimate of Emin. Each node also sets its estimate of Emin to the lower of its own energy-level and the lowest among the estimates heard from its neighbors. After a few iterations (intuitively, a number proportional to the network diameter), all nodes converge to the right Emin.

The goal of this paper is to examine secure data aggregation in sensor networks. Sensor networks have been pro-posed for scientific data collection, environmental monitoring, building health monitoring, burglar and fire alarm systems, and many other applications involving distributed interaction with the physical environment. Many of these applications involve a distributed system of sensors measuring the environment from many vantage points and then somehow aggregating the collected data to form a global summary view that can be acted upon. Consequently, data aggregation can be viewed as an important building block in sensor networks. Unfortunately, even though security has been identified as a major challenge for sensor networks, current proposals for data aggregation protocols have not been designed with security in mind, and consequently they are vulnerable to easy attacks. In this paper, we undertake an in-depth study of security for data aggregation in sensor networks. First, we show that existing proposals for data aggregation are subject to attack. When a single sensor node can be captured, compromised, or spoofed, an attacker can often manipulate the result of the aggregation operation without limit, gaining complete control over the computed aggregate. This is undesirable. For instance, we show that any protocol that computes the average, sum, minimum, or maximum function is insecure against malicious data, no matter how these functions are computed. In response to this threat, we introduce a theoretical frame- work for modeling the security of data aggregation. This model insists that the aggregation function must be resilient in the presence of arbitrary changes to a small subset of sensor observations, and thus we coin the term resilient aggregation to refer to schemes that satisfy this condition. We formalize this condition precisely and characterize which functions achieve the resilience condition. For instance, we show that the median is a more robust alternative to the average. Many of our conclusions are consistent with

intuition; the value of our mathematical framework is that one can place this intuition on a firm foundation, and one can analyze systems that are too complex for intuition to provide sufficient guidance.

Architecture:

A sensor network is a distributed system designed for interacting with the physical environment. A sensor network might contain hundreds or thousands of tiny, low-cost, low-power sensor nodes. Much architecture also uses more powerful base stations, which are in a one- to-many association with sensor nodes. Often, one forms a tree with a base station at the root and sensor nodes at the leaves. An aggregation transaction begins by broadcasting the query down the tree from the base station to the leaves. Then, the sensor nodes measure their environment and send their measurement back up the tree to the base station. Finally, the base station performs an aggregation computation to obtain the aggregate. Thus, sensor nodes act as data sources, and the base station acts as a sink.In this paper, we will abstract away some inessential features. We consider only a single base station and n associated sensor nodes. At some point, each sensor node takes a measurement and reports the observed value xi to the base station. The base station's goal is to compute an aggregate value y that summarizes the sensor readings x1; : : : ; xn using the aggregation function f; thus, y = f(x1; : : : ; xn). Our attacks and defenses ignore the inner workings of the specific aggregation and communication protocols used, such as how data is routed. Instead, we focus only on the function f computed by the base station. Consequently, we ignore the structure of the multi-hop network, assuming only that each sensor node has a separate link to the base station. We depict this abstracted architecture in Figure 1.

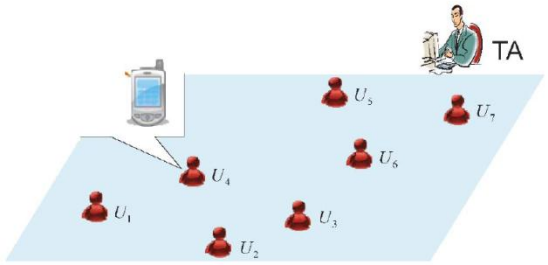

Fig: 1 System Architecture

Related Work:

Synopsis Count Diffusion:

In this module, each node generates a local synopsis which is a bit vector of length, where is the upper bound on Count. To generate, node executes the function, where X is the node's identifier. This module Algorithm can be interpreted as a coin-tossing with a cryptographic hash function, modeled as a random value whose output is 0 or 1, simulating a fair coin-toss, which returns the number of coin tosses, until the first head occurs or if tosses have occurred with no heads occurring. In the synopsis generation function, the ith bit of is set to "1" while all other bits are "0". Thus, it's a bit vector of the form with probability. The synopsis fusion function is the bitwise Boolean OR of the synopses being combined. Each node fuses its local synopsis with the synopsis it receives from other nodes. The final synopsis will be computed by BS by combining all of the synopses received from its child nodes. We compute that this will be a bit vector of certain

length. BS can estimate Count from via the synopsis evaluation function.

Synopsis Sum Diffusion

In this module the Count algorithm is extended for computing Sum. The synopsis generation function for Sum is a modification of that for Count, while the fusion function and the evaluation function for Sum are identical to those for Count. To generate the local synopsis to represent its sensed value, node invokes, uses Count Synopsis generation n times. In the ith, invocation, node executes the function where is constructed by concatenating its ID and integer value, and is the synopsis length. The value of is an upper bound on the value of Sum aggregate. Unlike the local synopsis of a node for Count, more than one bit in the local synopsis of a node for Sum may be equal to "1". The Count can be considered as a special case of Sum where each node's sensor reading is equal to one unit ions.

Threat Computation

In this module various attacks from unauthorized or compromised nodes are computed. To stop unauthorized nodes from interfering in communications among honest nodes, we can extend the aggregation framework with standard authentication and encryption protocols. However, cryptographic mechanisms cannot prevent attacks launched by compromised nodes because the adversary can obtain cryptographic keys from the compromised nodes. Compromised nodes might attempt to thwart the aggregate computation process in multiple ways. A compromised node which happens to be an in-network data aggregator may leak the sensor readings and sub aggregates which receives from child nodes. A compromised node can falsify its own sensor reading with the goal of influencing the aggregate value. If the local value of an honest node can be any value then a compromised node can pretend to sense any value. If the local value of an honest node is bounded, and a compromised node falsifies the local value within the bound, there is no solution for detecting such an attack. A compromised node can falsify the sub aggregate which is supposed to compute based on the messages received from child nodes.

Verification Computation

In this module we present a verification algorithm to detect the attacks. A compromised node launches the falsified sub aggregate attack by inserting one or more false "1"s in its fused synopsis. A straightforward solution to detect the falsified sub aggregate attack is the BS broadcasts an aggregation query message which includes a random value, Seed, associated to the current query. In the subsequent aggregation phase, along with the fused synopsis, each node also sends

MAC towards BS authenticating its sensed value. Node uses Seed and its own ID to compute its MAC. As a result, BS is able to detect any false "1" bits inserted in the final synopsis. In particular, if node contributes to bits in its local synopsis, it generates a MAC, where the key is that node shares with BS and the format of is Seed. Each node sends a message where might be needed by BS to regenerate the MAC for the verification. We observe that this approach requires MACs to be forwarded to BS, and hence, this approach is not suitable for a sensor network.

Algorithm Implementation

The main steps in the formal analysis of the correctness of an algorithm are:

Identification of the properties of input data (the so called problem's preconditions).

Identification of the properties which must be satisfied by the output data (the so-called

problem's post conditions).

Identification of the properties which must be satisfied by the output data (the so-called problem's post conditions).

Example. Let us consider the following algorithm whose aim is to find the minimal value in a finite

sequence of real numbers.

minimum(a[1::n])

min Ã a[1]

for i Ã 2; n do

if min > a[i] then min Ã a[i] endif

endfor

return min

The input array can be arbitrary, thus the preconditions set consists only of fn ¸ 1g (meaning

that the array is not empty).

The post condition is represented by the property of the minimal value: min · a[i] for all i = 1; n. We only have to prove that this post condition is satisfied after the algorithm's execution.

Thus, we will prove by using the mathematical induction that min · a[i], i = 1; n. Since

min = a[1] and the value of min is replaced only with a smaller one it follows that min · a[1].

Let us suppose that min · a[k] for all k = 1; i ¡ 1. Now, we only have to prove that min · a[i].

In the for loop, the value of min is modified as follows:

² If min · a[i] then min keeps its old value.

² If min > a[i] then the value of min is replaced with a[i].

Thus in both situations min will satisfy min · a[i]. According to the mathematical induction

method it follows that the post condition is satisfied, thus the algorithm is correct.

Let us consider now the following algorithm:

minim(a[1::n])

for i Ã 1; n ¡ 1 do

if a[i] < a[i + 1] then min Ã a[i]

else min Ã a[i + 1]

endfor

return min

Count Algorithm

In this algorithm, each node X generates a local synopsis Q^X which is a bit vector of length $\eta > \log N'$, where N' is the upper bound on Count. To generate Q^X, node X executes the function $CT(X, \eta)$ given as follows (Algorithm 1), where X is the node's identifier. Algorithm 1 can be interpreted as a coin-tossing experiment (with a cryptographic hash function $h()$, modeled as a random oracle whose output is 0 or 1, simulating a fair coin-toss), which returns the number of coin tosses, say i, until the first head occurs or $\eta + 1$ if η tosses have occurred with no heads occurring. In the synopsis generation function SG_{count}, the ith bit of Q^X is set to "1" while all other bits are "0". Thus, Q^X is a bit vector of the form $0^{(i-1)}10^{(\eta-i)}$ with probability 2^{-i}.

Algorithm 1 $CT(X, \eta)$

begin

i = 1;

while $i < \eta + 1$ *AND* $h(X, i) = 0$ **do**

i = i + 1;

end

return i;

end

Sum Algorithm

The Count algorithm can be extended for computing Sum. The synopsis generation function $SG()$ for Sum is a modification of that for Count, while the fusion function $SF()$ and the evaluation function $SE()$ for Sum are identical to those for Count.

To generate the local synopsis Q^X to represent its sensed value v_X, node X invokes $CT()$, used for Count synopsis generation, v_X times.[1] In the ith, $1 \leq i \leq v_X$ invocation, node X executes the function $CT(X_i, \eta)$ where X_i is constructed by concatenating its ID and integer i (i.e., $X_i = \langle X, i \rangle$), and η is the synopsis length. The value of η is taken as $\log_2 S' + 4$, where S' is an upper bound on the value of Sum aggregate. Unlike the local synopsis of a node for Count, more than one bit in the local synopsis of a node for Sum may be equal to "1". The pseudo code of the synopsis generation function, $SG_{\text{sum}}(X, v_X, \eta)$, is presented in Algorithm 2.

Algorithm 2 $SG_{\text{sum}}(X, v_X, \eta)$

begin

$Q^X[j] = 0 \; \forall j \; 1 \leq j \leq \eta$;

i = 1;

while $i \leq v_X$ **do**

$X_i = \langle X, i \rangle$;

$j = CT(X_i, \eta)$;

$Q^X[j] = 1$;

$i = i + 1$;

end

return Q^X;

end

System Environment

J2ME:

Java Platform, Micro Edition (Java ME) is the most ubiquitous application platform for mobile devices across the globe. It provides a robust, flexible environment for applications running on a broad range of other embedded devices, such as mobile phones, PDAs, TV set-top boxes, and printers. The Java ME platform includes flexible user interfaces, a

robust security model, a broad range of built-in network protocols, and extensive support for networked and offline applications that can be downloaded dynamically. Applications based on Java ME software are portable across a wide range of devices, yet leveraging each device's native capabilities.

The Java ME platform is deployed on billions of devices, supported by leading tool vendors, and used by companies worldwide. In short, it is the platform of choice for today's consumer and embedded devices. We are using J2ME in our application because we needed an environment which is adapted for devices which have extremely limited memory, small screen sizes, alternative input methods, slow processors. J2ME is Platform Independent and also provides high-level GUI features like Lists, Forms, Alerts, Textbox and Canvas.

J2ME is aimed squarely at consumer devices with limited horsepower. Many such devices (e.g., a mobile phone or pager) have no option to download and install software beyond what was configured during the manufacturing process. With the introduction of J2ME, "micro" devices no longer need to be "static" in nature. Not unlike a web browser downloading Java applets, an implementation of J2ME on a device affords the option to browse, download and install Java applications and content.

- Effective than Brew and Symbian
- Platform Independent
- Security
- Very simple

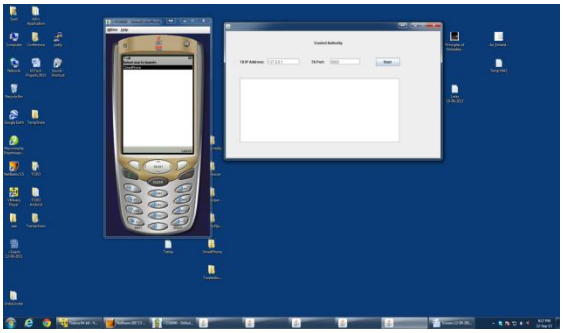

Fig 2:Start smart phone and trusted Authority

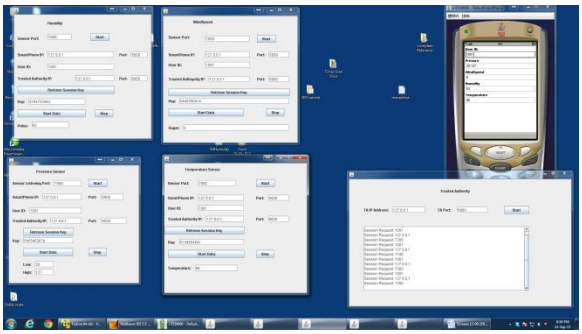

Fig 3:Start services in all instances and random key generated in each instance.

Conclusion:

We discussed the security issues of in-network aggregation algorithms to compute aggregates such as predicate Count and Sum. We discussed how a compromised node can corrupt the aggregate estimate of the base station,keeping our focus on the ring-based hierarchical aggregation algorithms. To address this problem, we presented a lightweight verification algorithm which would enable the base station (BS) to verify whether the computed aggregate was valid. For future work, we plan to design an efficient attack-resilient computation algorithm. This algorithm would guarantee the successful computation of the aggregate even in the presence of an attack.

REFERENCES

[1] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A secure hop-by-hop data aggregation protocol for sensor networks," in *Proc. Seventh ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006.

[2] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring sensor networks," in *Proc. 2nd Int. Workshop Sensor Network Protocols Applications*, 2003.

[3] L. Buttyan, P. Schaffer, and I. Vajda, "Resilient aggregation in sensor networks," in *Proc. 2nd IEEE Workshop Sensor Networks and Systems for Pervasive Computing*, 2006.

[4] S. Roy, S. Setia, and S. Jajodia, "Attack-resilient hierarchical data aggregation in sensor networks," in *Proc. ACM Workshop Security of Sensor and Adhoc Networks (SASN)*, 2006.

[5] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proc. IEEE Int. Conf. Data Engineering (ICDE)*, 2004.

[6] M. Garofalakis, J. M. Hellerstein, and P. Maniatis, "Proof sketches: Verifiable in-network aggregation," in *Proc. 23rd Int. Conf. Data Engineering (ICDE)*, 2007.

[7] L. Buttyan, P. Schaffer, and I. Vajda, "Resilient aggregation with attack detection in sensor networks," in *Proc. 2nd IEEE Workshop Sensor Networks and Systems for Pervasive Computing*, 2006.

[8] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proc. ACM Conf. Computer and Communications Security (CCS)*, 2006.

[9] M. B. Greenwald and S. Khanna, "Power-conservative computation of order-statistics over sensor networks," *Proc. 23th SIGMOD Principles of Database Systems (PODS)*, 2004.

[10] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *Proc. 2nd Int. Conf. Embedded Networked Sensor Systems (SenSys)*, 2004.

[11] H. Yu, "Secure and highly-available aggregation queries in large-scale sensor networks via set sampling," in *Proc. Int. Conf. Information Processing in Sensor Networks*, 2009.

[12] S. Nath, H. Yu, and H. Chan, "Secure outsourced aggregation via one-way chains," in *Proc. 35th SIGMOD Int. Conf. Management of Data*, 2009.

[13] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. F. Abdelzaher, "Pda: Privacy-preserving data aggregation in wireless sensor networks," in *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, 2007.

[14] K. B. Frikken and J. A. Dougherty, "An efficient integrity-preserving scheme for hierarchical sensor aggregation," in *Proc. 1st ACM Conf. Wireless Network Security (WiSec)*, 2008.