ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/262164880

PIP: Privacy and Integrity Preserving Data Aggregation in Wireless Sensor Networks

Conference Paper *in* Proceedings of the IEEE Symposium on Reliable Distributed Systems · September 2013

DOI: 10.1109/SRDS.2013.10

CITATIONS
20

READS
168

2 authors:

Vimal Kumar
The University of Waikato
30 PUBLICATIONS **384** CITATIONS

SEE PROFILE

Sanjay Kumar Madria
Missouri University of Science and Technology
398 PUBLICATIONS **3,863** CITATIONS

SEE PROFILE

All content following this page was uploaded by Vimal Kumar on 17 April 2018.

The user has requested enhancement of the downloaded file.

PIP: Privacy and Integrity Preserving Data Aggregation in Wireless Sensor Networks

Vimal Kumar and Sanjay Madria
Department of Computer Science
Misouri University of Science & Technology, Rolla, Missouri 65401
Email: vkq3f@mail.mst.edu, madrias@mst.edu

***Abstract*—With the exponential rise of pervasive computing applications, data privacy has become much more of an important issue than before. When data is aggregated at each hop in a sensor network, it becomes harder to protect its privacy. A number of privacy preserving data aggregation algorithms have recently appeared for wireless sensor networks (WSNs), very few of them however also address the issue of data integrity along with privacy. Data privacy and integrity are two contrasting objectives to achieve in general. In a privacy preserved data aggregation, it becomes easier for an attacker to inject false data hence, we suggest that both privacy and integrity of data should be treated together. In this paper, we present an energy efficient, privacy preserving data aggregation algorithm which also preserves data integrity in WSNs. We analyze the security of the algorithm and provide proofs for confidentiality and integrity. We enhance this algorithm further to localize, to a certain degree, the corrupt aggregator. We provide the results of our implementation of the algorithm on TelosB motes, illustrating that both the computational overhead and the energy consumption are very low. Finally, we compare our algorithm with other schemes having similar objectives demonstrating that our algorithm performs better in terms of bandwith usage and energy consumption in a WSN environment.**

***Keywords*-Wireless sensor networks; Data aggregation; Secret sharing;**

I. Introduction

A wireless sensor network (WSN) consists of a large number of small, low cost, battery operated, wireless sensors. WSNs have a multitude of applications such as environmental sensing, health-care monitoring, traffic monitoring, law enforcement, and military, among others. In-network processing techniques such as sensor data aggregation are used to reduce communication overhead in WSNs. Data aggregation refers to using an aggregation function f such as SUM, AVG, MIN/MAX, MULT, etc. on n data items, at intermediate nodes, to produce one aggregate value $v = f(x_1, x_2, ..., x_n)$. Aggregating data on intermediate nodes not only reduces the energy consumption, but also minimizes end to end delay and bandwidth usage as well as increase the throughput of the network.

Increasing proliferation of wireless sensors in our daily lives has given rise to the need of privacy preserving and secure data aggregation. An excellent motivating example of the application of privacy preserving data aggregation was provided in [1] which we reproduce with a modification for the need of integrity. *A utility company may want to monitor the aggregate electricity and water consumption of a neighborhood by placing wireless sensors in the houses. This aggregate information may be helpful for the company and the community however individuals run the risk of their privacy being exposed as the data is communicated via multiple hops to the base station. This private information may reveal a lot about day to day activities of a household and could be used maliciously; hence individuals may only be willing to participate if the privacy of their data is guaranteed [1]. In addition to this, malicious insider nodes may also inject false data in the aggregate for personal benefits and hence it is also required that integrity of data be preserved.*

In this paper, we address the issues of *privacy* and *integrity* in data aggregation in wireless sensor networks. By *privacy* we essentially mean data privacy and thus it pertains to hiding the details of a node's data from other nodes in the network. *Integrity* refers to preventing the modification of data by corrupt aggregators. *Privacy* is typically obtained by techniques (e.g., encryption) that produce output resembling random bits. On the contrary, *integrity* is usually obtained by peer monitoring, which does not work with randomized data. Our work differs from the work of many others in that we provide a scheme that combines these two seemingly contrasting objectives of *privacy* and *integrity*.

II. Problem Statement

We consider a WSN with n nodes, organized in a tree hierarchy, and rooted at the base station *(B.S)*. Each sensor S_i generates a reading δ_i, the aggregate of k such readings is $\sum_{i=1}^{k} \delta_i$. Our definition of privacy is based on the well-known concept of semantic security, that the adversary cannot determine any information about the plaintext from a given ciphertext in polynomial time. We define the set of nodes which have access to the sensor reading δ_i as the privacy set Π_i. A sensor reading is private iff its privacy set consists of the node generating the reading itself and the base station. Similarly, an aggregate is private iff its privacy

set consists of only the base station. Thus, we define the following conditions for privacy preserved data aggregation.

- For δ_i to be private, $\Pi_i = \{S_i, B.S\}$
- For $\sum_{i=1}^{k} \delta_i$ to be private, $\Pi_{1...k} = \{B.S\}$

If *k* out of *n* nodes participate in data collection and each node S_i contributes a reading δ_i, the aggregate is $\sum_{i=1}^{k} \delta_i$. A data integrity violation occurs if the aggregate received by the base station $\sum_{i=1}^{k} \delta'_i \neq \sum_{i=1}^{k} \delta_i$.

Most works on privacy preserving data aggregation focus solely on privacy while completely ignoring the data integrity aspect. Privacy preserving data aggregation enables an aggregator to aggregate *k* data items without knowing what they are. This leaves the aggregation operation prone to false data injection attacks by corrupt aggregators. This can be overcome by using aggregate signatures such as in [2]. Signing and verification however, have a high energy cost and our requirement is of an algorithm which can provide integrity of the private data while also being energy efficient. Based on the above discussion, we define our objectives as below:

- To present a scheme for energy efficient privacy preserving data aggregation
- To provide energy efficient mechanism for the integrity of data aggregated privately.
- Simulation and implementation of the algorithm on real motes to verify its performance and energy usage.

III. Related Work

A number of privacy preserving algorithms for data aggregation in WSNs have been proposed. Some of them, such as [2] and [3] make use of asymmetric homomorphic encryption to provide privacy of aggregate data. Asymmetric homomorphic cryptography provides the required privacy but also consumes a great deal of energy as shown in the comparison provided later in section VIII. This approach may not always be viable on energy constrained wireless sensors and thus, there is a need for more efficient solutions.

Two privacy preserving algorithms, CPDA and SMART are proposed in [1]. In CPDA, nodes hide their data in a random polynomial and send it to all other nodes in the cluster using pair wise keys. These polynomials are added up and sent to the cluster head, along with a secret. The authors then discuss SMART which is less communication intensive than CPDA. In SMART each node in an aggregation tree slices its data into a fixed number of parts and sends each part to a neighboring node, keeping one for itself. Nodes add all of the received slices together and send them to the aggregator along the path of the tree. In CPDA, the aggregate is revealed to the cluster head while, in SMART, because positive slices of data are distributed, some amount of information about the data is always leaked. Moreover the privacy of data depends on key pre-distribution. If an adversary is able to capture a sufficient number of nodes, the privacy in both schemes suffers heavily. The scheme in [4] utilizes secret perturbation to address the issue of privacy. This scheme prevents the aggregator from knowing the aggregate, and no privacy is lost when a few nodes are compromised. In PASKOS and PASKIS algorithms proposed in [5] each node is assigned a subset of keys from a key ring. In PASKOS, nodes use all of their keys to create hashes, and randomly either add or subtract them from the data. This randomized data and a list indicating whether the hashes were added or subtracted is sent to the aggregator. The authors further improve the algorithm such that even the base station is oblivious to the aggregate in the PASKIS protocol. These protocols protect the aggregate, even from the intermediate nodes. As in [1] however, the privacy depends on key pre-distribution. Moreover, because a list, whose size is the size of the key ring, must be sent by every node, the bandwidth consumption of the protocols is also very high. In [6] each node establishes a twin key with at least one other node in a cluster. The twin key is used to either add or remove data from the aggregate. Each node, prior to sending data to the next node, creates a shadow value, encrypts it with the twin key, and adds it to the aggregate. Twin key has the property that if used again it will cancel out the effect of the first use. Thus, when a shadow is added to the aggregate, a node sharing that twin key will remove the shadow when the aggregate comes to it during aggregation. This scheme again suffers from the problem that a sufficient number of captured nodes can reveal the privacy of a number of other nodes. [7] proposed a novel solution for multi-dimensional data aggregation using a bilinear pairing based public key encryption. This scheme saves bandwidth when n-dimensional data has to be aggregated. The downside, however, of this scheme is that it is not very energy efficient.

Schemes in [1] [4] [5] [6] [7] address the issue of privacy of data. They do not however, provide the integrity of data. In many cases, the goal of the adversary is to have the base station accept false readings which these schemes fail to protect against. Schemes in [8] [9] [10] were proposed to handle an adversary that tries to inject false data in the system. iPDA [9] handles data integrity along with the privacy of data. It uses the SMART [1] algorithm and preserves the integrity of data by using two parallel but disjoint aggregation trees rooted at the base station. The algorithm assumes that it will be difficult for an attacker to inject data and affect both the aggregation trees equally. An adversary however, which has captured at least two nodes, one in each tree, can easily subvert this scheme. iCPDA [10] uses peer monitoring to protect integrity of data. In this scheme, the privacy of individual sensor data is protected, while the privacy of partial aggregates is not. Moreover, an adversary can simply eavesdrop on the clear-text communication of nodes and obtain the aggregate. The aggregators are assumed to be trustworthy and aggregates

are revealed to them. If a node determines an aggregator is misbehaving, it sends a report to the base station. The base station, however, faces the dilemma of whether or not to trust the reporting node. The scheme proposed in [8] also provides both integrity and privacy. In this scheme, secrets are pre-deployed on intermediate nodes to assess the integrity of the data. The scheme is based on random perturbation and therefore is very energy efficient. Though, it can only provide an approximation of the data as it is based on histogram of the sensor data. Unlike [8] our scheme provides accurate data aggregation. Our goals are similar to the goals in [9] and [10], i.e. the privacy and integrity of data. Additionally, our algorithm does not reveal partial aggregates to intermediate nodes. Due to the similarity of goals we will compare our privacy and integrity preserving algorithm PIP to iPDA [9] and iCPDA [10].

IV. System and Adversary Models

Our network consists of n wireless sensors deployed in the field and a base station. Wireless sensors have limited energy and communicate in a small radius and therefore, resort to hop-by-hop communication. The base station has unlimited energy and is secure. Each sensor S_i generates a reading δ_i the aggregate, of which needs to be sent to the base station. Aggregator nodes perform data aggregation, and the base station receives $\sum \delta_i$. We assume that the adversary only tries to corrupt the aggregators and not the leaf nodes, as a corrupt aggregator will have a larger impact on the aggregated value. We limit ourselves to the SUM aggregation function in this paper.

Our goal is to provide privacy and integrity preserving aggregation of data. Our algorithm defends against an adversary who will attempt to.

- Compromise the data privacy of the aggregators in the network.
  In this case, for a sensor reading δ_i of a node S_i, the privacy set $\Pi_i = \{S_i, B.S, S_i, ..., S_{i+p}\}$ or for an aggregate $\sum_{i=1}^{k} \delta_i$, $\Pi_{1...k} = \{B.S, S_i, ..., S_{i+p}\}$, where the data is accessible to nodes $S_i, ..., S_{i+p}$.
- Inject false data into the network.
  In this case the aggregate data received by the base station, $\sum_{i=1}^{k} \delta'_i$ and the actual aggregate data $\sum_{i=1}^{k} \delta_i$ do not match i.e $\sum_{i=1}^{k} \delta'_i \neq \sum_{i=1}^{k} \delta_i$.

V. Preliminaries

A. Secret Sharing

In secret sharing, a data item δ is divided into n parts such that any k parts can be used to reconstruct the data and the knowledge of $k-1$ parts does not provide any information.

To create shares of a data item δ using Shamir's [11] scheme, a prime number p is chosen such that $p > max(\delta, n)$. A random $k-1$ degree polynomial is then created, where the coefficients $a_1, ..., a_{k-1}$ are chosen randomly from a uniform distribution over the integers in $[0, p]$ and $a_0 = \delta$. This polynomial $\omega(x) = a_0x_0 + a_1x_1 + ... + a_{k-1}x_{k-1}$ is then sampled at points $x = 1, ..., n$ to create shares $\omega(1), ..., \omega(n)$.

Any k out of the n shares created above can be used to create a $k-1$ degree polynomial $\omega'(x)$ using polynomial interpolation. To reconstruct the original data $\omega'(x)$ is evaluated at $x = 0$.

B. Recursive Secret Sharing

Parakh and Kak [12] proposed a recursive secret sharing scheme based on Shamir's secret sharing [11]. Recursive secret sharing is proposed for efficient storage, where the shares of the data item δ are used to store $k-2$ additional pieces of information. A node with all the shares can easily reconstruct δ and the $k-2$ pieces of hidden information.

We denote recursive secret sharing by RSS and Shamir's secret sharing by SSS, respectively. In RSS Parakh and Kak, modify SSS such that the shares are generated using polynomial interpolation rather than by generating random polynomial and sampling it. For example, suppose secrets $\delta_1, ..., \delta_{k-1}$ are to be stored in the shares. A random number y_{11} is chosen and using $(0, \delta_1)$ and $(1, y_{11})$, a 1-degree polynomial $\omega^1(x)$ is interpolated. This polynomial $\omega^1(x)$ is then sampled at $x = 2$ and $x = 3$ to obtain $\omega^1(2)$, $\omega^1(3)$. Next, $(0, \delta_2)$ and the points $(1, \omega^1(2))$ and$(2, \omega^1(3))$ are used to interpolate a 2-degree polynomial $\omega^2(x)$. The polynomial $\omega^2(x)$ is sampled at $x = 3, 4$ and 5 to obtain $\omega^2(3), \omega^2(4)$ and $\omega^2(5)$. These three points are used as y coordinates for $x = 1, 2, 3$ along with the point $(0, \delta_3)$ to interpolate the 3-degree polynomial $\omega^3(x)$. This process is repeated until the last secret δ_{k-1} is used to create a $k-1$ degree polynomial $\omega^{k-1}(x)$. This polynomial can then be sampled at n points to creates n shares, any k of which can be used to regenerate δ_{k-1}. In the reconstruction phase, δ_{k-1} is regenerated using the original method discussed in the previous subsection by constructing $\omega^{k-1}(x)$ using the k shares. This polynomial $\omega^{k-1}(x)$ is then sampled at $x = 1, ..., k-1$ to obtain $\omega^{k-1}(1), ..., \omega^{k-1}(k-1)$. The points $(k-1, \omega^{k-1}(1))$,...,$(2(k-1), \omega^{k-1}(k-1))$ are then used to interpolate a $k-2$ degree polynomial $\omega^{k-2}(x)$. δ_{k-2} is recovered by evaluating $\omega^{k-2}(x)$ at $x = 0$. This process is repeated for all δ_i until secret δ_1 is recovered.

VI. Proposed Algorithm

In this section, we present PIP: Privacy and Integrity Preserving Data Aggregation Algorithm, which fulfills the objectives stated in section II. We begin by addressing the energy efficient privacy and integrity preservation objectives. We base our algorithm on the recursive secret sharing algorithm described in section V-B. The authors in [12] proposed the scheme for efficient storage, where the shares of the data δ are used to store $k-2$ additional pieces of information. A node with at least k shares can easily reconstruct all of the $k-1$ pieces of hidden information. We

provide a construction in which we can prevent a node which has all the shares from reconstructing and retrieving the hidden data. Our construction preserves the homomorphic property of SSS and RSS which is used to aggregate data in a privacy preserving manner.

The basic methodology is to create shares for a given sensor reading using RSS. Each node then sends all of the shares to its parent. Because RSS is based on SSS, it is additively homomorphic and so the parent node will be able to aggregate the shares to aggregate the data. However, because each node is sending all the shares to its parent, it is straightforward for a curious aggregator to regenerate the data and thus subvert its privacy. To prevent this, each node scrambles the shares by subtracting modulo prime, a scrambling key shared between each node and the base station from the shares. This scrambling procedure is very light weight and preserves the homomorphic property of RSS. We could also use homomorphic encryption here. Scrambling however, is preferred since it consumes negligible energy when compared to encryption while still being secure, as will be seen in the security proof later. Finally, because we also want to preserve the integrity of the data, the intuition is to have some additional information, along with the shares which will help the base station to detect any changes in the data. We use an integrity key, which each node shares with the base station and embed this key in the generated shares using RSS. It should be noted that Verifiable Secret Sharing (VSS) could also be used to ensure data integrity. VSS schemes however, typically require energy intensive cryptographic constructs and are not efficient communication wise. Our scheme as we show in section VIII is both computationally light weight and efficient in terms of communication overhead.

A. Basic PIP Algorithm

Basic PIP consists of two algorithms, one for generating the shares from the data and the keys and the other for regenerating data and checking its integrity. We assume that the sensor nodes are randomly deployed in a field and they self organize into a tree hierarchy using the tree construction algorithm given in [2]. The base station preloads each sensor with a pseudo random function (PRF) $f(.)$, a network wide nonce n, three distinct random seeds r_1, r_2, r_3, a network-wide random seed r_4 and the network-wide prime number p. Before every iteration, a sensor node uses the PRF $f(.)$ on the nonce n, using the random seeds r_1, r_2, r_3, r_4 as keys, to generate the perturbation key η, a scrambling key Ψ and an integrity key $I = \{I', I''\}$ as shown in Algorithm 1. The newly generated keys will be used as random seeds for the next iteration. All the operations are performed mod p. In the basic PIP algorithm, a sensor S_k uses the perturbation key η_k, and the sensor reading δ_k to generate perturbed data $\delta_{\eta k}$. Recursive secret sharing is then used and a linear combination of the integrity key $I_k^{'} + I_k^{''}\delta_{\eta k}$ and data, $\delta_{\eta k}$ are encoded to create the shares $\omega_k^2(3), \omega_k^2(4)$, and $\omega_k^2(5)$. If these shares are sent to the aggregator, then a malicious aggregator would be able to easily replace the data in the shares with corrupt data, keeping the integrity key intact. We use the scrambling key Ψ_k to scramble the shares and prevent this from happening. The scrambled shares λ_k^1, λ_k^2, and λ_k^3 are generated as shown in the algorithm. Each leaf node sends the scrambled shares λ_k^1, λ_k^2, and λ_k^3 to its parent, which aggregates the shares received from all of its children. The aggregate shares $\lambda_{agg}^1, \lambda_{agg}^2$, and λ_{agg}^3 are calculated as $\lambda_{agg}^1 = \sum \lambda^1, \lambda_{agg}^2 = \sum \lambda^2, \lambda_{agg}^3 = \sum \lambda^3$, where the summation is over all children. The base station then uses the sum of the scrambling keys $\sum \Psi$ to unscramble the aggregate shares to obtain $\omega_{agg}^2(3), \omega_{agg}^2(4)$ and $\omega_{agg}^2(5)$. These shares are then used to decode received perturbed data $\omega_{agg}^1(0)$ and the linear combination of integrity keys and the perturbed data $\omega_{agg}^2(0)$. If the equation $\omega_{agg}^2(0) - \omega_{agg}^1(0)I_k^{''} = \sum_{k=1}^{n} I_k^{'}$ holds, data has not been tampered with and is recovered after removing the perturbation. Otherwise, the data has been tampered with and the base station would need to take appropriate steps. Algorithmic details of the share verification and data regeneration process can be seen in Algorithm 2.

Algorithm 1 Share Generation Algorithm

Require: Sensor reading δ_k, PRF $f(.)$, nonce n, random seeds r_1, r_2, r_3, r_4.

1: Generate Perturbation key as $\eta_k = f_{r_1}(n)$, Scrambling key $\Psi_k = f_{r_2}(n)$, Integrity key $I_k = \{I_k^{'}, I_k^{''}\} = \{f_{r_3}(n), f_{r_4}(n)\}$.
2: Update $r_1 = \eta_k$, $r_2 = \Psi_k$, $r_3 = I_k^{'}$, $r_4 = I_k^{''}$.
3: Generate perturbed data $\delta_{\eta k} = \delta_k + \eta_k$.
4: Choose a random number y uniformly from $\mathbb{Z}_p$.
5: Interpolate $(0, \delta_{\eta k})$ and $(1, y)$ to obtain a degree 1 polynomial ω_k^1.
6: Sample the polynomial ω_k^1 at $x = 2, 3$ and interpolate $(0, I_k^{'} + I_k^{''}\delta_{\eta k}), (1, \omega_k^1(2)), (2, \omega_k^1(3))$ to obtain a degree 2 polynomial ω_k^2.
7: Sample ω_k^1 at $x = 3, 4, 5$ to obtain $(3, \omega_k^2(3)), (4, \omega_k^2(4)), (5, \omega_k^2(5))$.
8: $\omega_k^2(3), \omega_k^2(4), \omega_k^2(5)$ are the shares which consist of the sensor reading δ_k and the integrity key I_k of the sensor k.
9: These shares are then scrambled using the scrambling key Ψ_k as follows.
$\lambda_k^1 = \Psi_k - \omega_k^2(3)$
$\lambda_k^2 = \omega_k^2(3) - \omega_k^2(4)$
$\lambda_k^3 = \omega_k^2(4) - \omega_k^2(5)$
10: Final shares are the scrambled shares $\lambda_k^1, \lambda_k^2, \lambda_k^3$.

B. Numerical example

We provide a working example of the basic PIP algorithm below. Let us consider a small aggregation tree consisting of 3 nodes, as shown in Fig. 1. Node A aggregates the data from nodes B, C and itself and sends the aggregate to the base station. Each node generates the scrambling key Ψ, the integrity key $I = \{I', I''\}$ and the perturbation

Algorithm 2 Data Regeneration and Integrity Check

Require: PRF $f(.)$, nonce n, random seeds r_1, r_2, r_3, r_4.

1: B.S generates the scrambling, perturbation and the integrity keys for all the sensors and computes $\sum_{k=1}^{n} \Psi_k$, $\sum_{k=1}^{n} \eta_k$ and $\sum_{k=1}^{n} I_k^{'}$.

2: Root node unscrambles the shares using $\sum_{k=1}^{n} \Psi_k$ as follows.
$\omega_{agg}^2(3) = \sum_{k=1}^{n} \Psi_k - \lambda_{agg}^1$
$\omega_{agg}^2(4) = \omega_{agg}^2(3) - \lambda_{agg}^2$
$\omega_{agg}^2(5) = \omega_{agg}^2(4) - \lambda_{agg}^3$

3: Interpolate $(3, \omega_{agg}^2(3)), (4, \omega_{agg}^2(4))$ and $(5, \omega_{agg}^2(5))$ to obtain a degree 2 polynomial ω_{agg}^2.

4: Sample ω_{agg}^2 at $x = 1, 2$.

5: Interpolate $(2, \omega_{agg}^2(1)), (3, \omega_{agg}^2(2))$ to obtain a degree 1 polynomial ω_{agg}^1.

6: Calculate $\omega_{agg}^1(0)$ and $\omega_{agg}^2(0)$

7: **if** $\omega_{agg}^2(0) - \omega_{agg}^1(0) I_k^{''} == \sum_{k=1}^{n} I_k^{'}$ **then**
Accept the aggregate as $\omega_{agg}^1(0) - \sum_{k=1}^{n} \eta_k$.

8: **else**
Aggregate is corrupt.

9: **end if**

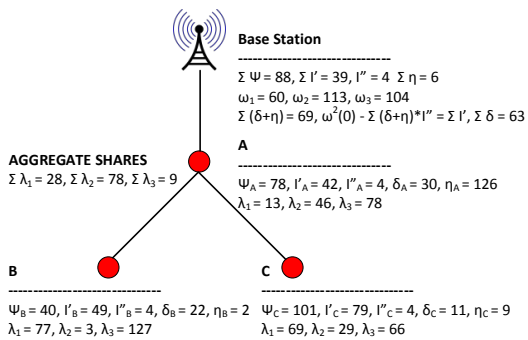

Figure 1. Data aggregation using Basic PIP

key η. Using these keys, the nodes sense data, perturb it and generate scrambled shares. To illustrate the share generation process, let us take node B as an example, which has the scrambling key $\Psi_B = 40$, the integrity key $I_B = 49, 4$ and the perturbation key $\eta_B = 2$. All calculations are done mod 131, which is a network parameter. Node B has data $\delta_B = 22$, to which it adds the perturbation key $\eta_B = 2$ and generates perturbed data $\delta_{\eta B} = 24$. It chooses a random number $y = 7$ and interpolates $(0, 24)$ and $(1, 7)$ to generate the polynomial $\omega_B^1(x) = 114x + 24$. This polynomial is sampled at $x = 2$ and 3 to obtain the points $(2, 121), (3, 104)$. Next we compute $I_B^{'} + I_B^{''}\delta_{\eta B}$ = 14 and interpolate $(0, 14), (1, 121)$ and $(2, 104)$ to generate the polynomial $\omega_B^2(x) = 69x^2 + 38x + 14$. We sample this polynomial at $x = 3, 4, 5$ to generate the recursive shares $94, 91$, and 95. We then use the scrambling key $\Psi_B = 40$ to obtain the final scrambled shares $\lambda_B^1 = 77, \lambda_B^2 = 3$ and $\lambda_B^3 = 127$. Node B sends it's scrambled shares to node A, as does node C. Node A, being the aggregator, aggregates its own shares with the shares it receives to obtain the aggregate shares $\sum \lambda_{agg}^1 = 28, \sum \lambda_{agg}^2 = 78, \sum \lambda_{agg}^3 = 9$. The aggregate shares are then sent to the base station. Because the keying material of all the nodes is shared with the base station, the base station can calculate $\sum \Psi$, $\sum I'$, I'' and $\sum \eta$ which, in this case, are 88, 39, 4 and 6 respectively. The base station then unscrambles the aggregate shares using $\sum \Psi$. The original shares after unscrambling come out to be 60, 113, and 104. These shares are interpolated to obtain the polynomial $\omega_{agg}^2(x) = 100x^2 + 8x + 53$. The polynomial $\omega_{agg}^2(x) = 100x^2 + 8x + 53$ is then sampled at $x = 1, 2$ to obtain points $(1, 30)$ and $(2, 76)$ respectively. Points $(2, 30)$ and $(3, 76)$ are then interpolated, and the polynomial $\omega_{agg}^1(x) = 46x{+}69$ is obtained. The base station then checks whether the equation $\omega_{agg}^2(0) - \omega_{agg}^1(0) I_k^{''} = \sum_{k=1}^{n} I_k^{'}$ holds. Since we can see that the equation is satisfied in the example, $\omega_{agg}^1(0) = 69$ is taken as the sum of the perturbed data $\sum \delta_\eta$ from which $\sum \delta$ can be derived as $\sum \delta = \omega_{agg}^1(0) - \sum \eta$.

C. Security Analysis

1) Confidentiality: The basic PIP algorithm can be reduced to the CMT scheme described in [13], which is proven to be semantically secure. The CMT scheme is based on pseudo random functions (PRFs). Given a nonce *r*, a data point $\delta \in$ [0, M-1], where M is an integer, a PRF *f(.)*, an encryption key k and a length matching function *h(.)*, the CMT scheme encrypts the data point δ as below.

$$Enc_k(\delta) = (\delta + h(f_k(r)))mod\ \mathrm{M} \tag{1}$$

The basic PIP scheme can be stated formally as below. Given an input of a nonce *r*, a data point $\delta \in$ [0, p-1], where *p* is a large prime , the scrambling key Ψ and the perturbation key η, the scheme first generates perturbed data.

$$\delta_\eta = (\delta + \eta)mod\ p \tag{2}$$

The scheme then uses the RSS(.) function to create the shares. This can be represented as

$$RSS(\delta_\eta, f_k(r)) = \lambda_1, \lambda_2, \lambda_3 \tag{3}$$

Using $f_k(r)$ gives the shares a non-deterministic character. Using a PRF with a nonce and a sensor specific key is the practical way of generating the random number *y* in step 2 of Algorithm 1. We see from Algorithm 2 that if the first share is correctly recovered, we can recover the other shares easily. Thus the security of the scheme hinges on the security of the first share. We define a function *g(.)*, such that

$$g(\lambda_1) = -\lambda_1 \tag{4}$$

then, using equations (2), (3) and (4), the basic PIP scheme can be written as

$$PIP_k(\delta) = (\Psi + g(\lambda_1))mod\ p \tag{5}$$

Equation (5) is in the same form as (equation 1), where the randomness generated by $f_k(r)$ in equation (1) is generated by $g(\lambda_1)$ in equation (5). By this reduction, we can say that the basic PIP scheme is at least as secure as the CMT scheme in [13].

2) Integrity: We say that a scheme protects the integrity of a message if the probability of an adversary performing existential forgery is negligible. The integrity of the basic PIP scheme is based upon its confidentiality, since the integrity key is wrapped in the shares along with the perturbed data. An adversary that is able to break the confidentiality of the scheme will also be able to break the integrity of the scheme. Thus, we say that, for every adversary A attacking the basic PIP scheme for integrity, $\exists$ an adversary B, attacking its confidentiality. For an adversary that possesses messages δ_0 and δ_1 and is given Λ_b, where $b \in 0, 1$(From here on, for the sake of convenience we denote the scrambled shares of node k by Λ_k), we define a PPT distinguisher $D(\Lambda_0, \Lambda_1, \delta_0, \delta_1)$ which outputs 1 if it is able to distinguish between the two Λs. The advantage of the adversary can then be written as

$$Adv_{PIP_C}[B] = Pr[D(\Lambda_0, \Lambda_1, \delta_0, \delta_1) = 1] \quad (6)$$

Using (6) the advantage of an adversary attacking the integrity of the scheme can be written as

$$Adv_{PIP_I}[A] \leq Adv_{PIP_C}[B] + (1/2^{|\Psi|} * 1/2^{|\eta|}) \quad (7)$$

In (7), the first term corresponds to the adversary attacking the confidentiality of the scheme. The second term corresponds to an attacker randomly guessing the keys Ψ and η, where $|\Psi|$ and $|\eta|$ are the respective key sizes. We showed earlier that the scheme is semantically secure. The term $Adv_{PIP_C}[B]$ is therefore negligible.

$$Adv_{PIP_I}[A] \leq 1/2^{|\Psi|} * 1/2^{|\eta|} \quad (8)$$

$$Adv_{PIP_I}[A] \leq 1/2^{2b} \quad (9)$$

where b is the size of the shares and $b = |\Psi| = |\eta|$. We can determine from (9) that the advantage of the adversary attacking the integrity is negligible and the scheme protects the integrity of the data as long as b (the share size) is sufficiently large.

D. Discussion on Privacy and Integrity in Basic PIP

In semantic security, an adversary in possession of Λ_1 and Λ_2 will be unable to differentiate between the two. Thus, with shares of size b, there are 2^b equally likely possibilities and no information is leaked to the aggregator regarding either the original shares or data. Thus, in basic PIP, for sensor S_i, which has data δ_i, the privacy set consists only of S_i, however in the following section on enhanced PIP, we will see that a node may have to reveal its shares to the base station for verification, thus $\Pi_i = \{S_i, B.S\}$. The same reasoning can be applied to aggregate data of k nodes. An aggregator receives private data from $k-1$ nodes, and aggregates the scrambled shares. Again, there are 2^b equally likely possible original aggregate shares. Each node however, only knows its own Ψ and so the knowledge of aggregate scrambled shares would not be enough to gain any information regarding the aggregate data. In this case $\Pi_{i \in Agg} = \{B.S\}$, where Agg is the set of nodes which took part in data aggregation.

In recursive secret sharing, if k shares are known, original aggregate $\sum \delta_\eta$ can be replaced by false data $\sum \hat{\delta}_\eta$ in the shares without altering the integrity key. However, we established above that a node not in possession of the scrambling key Ψ cannot obtain the original shares. This means that, to inject false data into the system, a node will have to change the scrambled shares. Changing the scrambled shares, amounts to changing all the embedded data in the shares. In this case the embedded data is the aggregate data $\sum \delta_\eta$ and $\sum I' + I'' \sum \delta_\eta$. Thus we conclude that in recursive shares $\Delta \sum \delta \cong \Delta I$. This means that if the B.S discovers that $\sum I' \neq \sum \hat{I}'$, where $\sum \hat{I}'$ is the sum of the integrity keys $\hat{I}'$ derived from the received shares, it is equivalent to $\sum \delta_\eta \neq \sum \hat{\delta}_\eta$.

VII. Extensions to Basic PIP algorithm

A. Enhanced PIP for Corrupt Aggregator Detection

The basic PIP algorithm allows us to determine whether or not the aggregate data is corrupt. It, however, does not enable us to detect which malicious aggregators injected false data. To identify such malicious aggregators we perform a two fold check in the enhanced PIP algorithm. To begin with, we make an additional assumption that the base station can reach all the nodes in the network using a secure channel. As before, we assume that each sensor node is preloaded with a pseudo random function (PRF) $f(.)$, a network wide nonce n, three distinct random seeds r_1, r_2, r_3 and a network-wide random seed r_4 which are used to generate the scrambling, integrity and the perturbation keys along with the network-wide prime number p. This time however, the nodes are provided with another random seed r_5 to generate an additional scrambling key such that the nodes have two scrambling keys Ψ^1 and Ψ^2. The network hierarchy is known to the base station and hence it can calculate $\sum \Psi^2$ for any aggregator in the network. Additionally, each parent child pair (S_p, S_k) shares a commitment key γ_{pk}. Leaf nodes begin the data collection by generating the scrambled shares through the process shown in Algorithm 1, using the scrambling key Ψ^1. The leaf nodes then recursively encode the commitment key γ_{pk}, (shared with their parent node) into the scrambled shares. These shares are then scrambled once again using the scrambling key Ψ^2 before being sent to the parent. The parent node receives Λ from all of its children and computes $\Lambda_{agg} = \sum \Lambda$. It then computes $\sum \gamma_{pc}$, the sum of the commitment keys it shares with the children that sent their Λs. It also calculates its own scrambled shares Λ_p

Algorithm 3 Data Aggregation using enhanced PIP

Require: Sensor reading δ_k, PRF $f(.)$, nonce n, random seeds r_1, r_2, r_3, r_4, r_5, commitment keys γ_{kc} and γ_{pk} shared with the child and parent nodes respectively.

1: Generate Perturbation key $\eta_k = f_{r_1}(n)$, Scrambling keys $\Psi_k^1 = f_{r_2}(n)$, $\Psi_k^2 = f_{r_2}(n)$, Integrity key $I_k = \{I_k^{'}, I_k^{''}\} = \{f_{r_3}(n), f_{r_4}(n)\}$.

2: Update $r_1 = \eta_k$, $r_2 = \Psi_k^1$, $r_3 = I_k^{'}$, $r_4 = I_k^{''}$, $r_5 = \Psi_k^2$.

3: Generate scrambled shares $\lambda_k^1, \lambda_k^2, \lambda_k^3$ using Algorithm 1 with Ψ_k^1 as the scrambling key.

4: Leaf nodes interpolate $(0, \gamma_{pk}), (1, \lambda_k^1), (2, \lambda_k^2), (3, \lambda_k^3)$ to generate the polynomial $\omega_k^3(x)$ and then sample it to obtain shares $\omega_k^3(4), \omega_k^3(5), \omega_k^3(6), \omega_k^3(7)$, which are then scrambled using Ψ_k^2 to obtain final scrambled shares Λ_k.

5: Parent nodes aggregate the shares sent by their children and compute $\sum \Lambda_{agg}$.

6: Parent nodes compute $\sum_{i=0}^{j} \gamma_{kc_i}$, s.t. j is the number of children and γ_{kc_i} is the key shared with child i .

7: Parent nodes use $\gamma_{pk} - \sum_{i=0}^{j} \gamma_{kc_i}$ as the commitment key and generate their scrambled shares Λ_k. $\sum \Lambda_{agg}$ and Λ_k are then sent to the grand parent node.

Algorithm 4 Corrupt aggregator detection

Require: Corrupt shares $\sum_{k=1}^{j} \Lambda_k$, Ψ^2 for all sensor nodes.

1: Set $R = \{\phi\}$

2: Base station queries the node S_{ca} from which corrupt aggregate was received for its commitment keys shared with the children and Λ, Λ_c received from the children, where $c \in C$ and C is the set of the child nodes.

3: **repeat**

4: Base station chooses a previously unselected child node S_i, computes appropriate $\sum \Psi^2$, and decodes the commitment key $\gamma_{kc_i}^{'}$

5: **if** $\gamma_{kc_i}^{'} \neq \gamma_{kc_i}$ **then**
Perform step 1 for child i.

6: **if** $\Lambda \neq \Lambda_i$ **then**
$R = \{S_{ca}, S_i\}$.

7: **else**
Commitment keys are checked for all children of the child node.

8: **if** All keys match **then**
$R = \{S_{ca}\}$.

9: **else**
Go to step 5.

10: **end if**

11: **end if**

12: **end if**

13: **until** All child nodes in C are selected.

14: Output R as the set of corrupt nodes.

using both the scrambling keys Ψ^1 and Ψ^2 and $\gamma_{gp} - \sum \gamma_{pc}$ as the commitment key. γ_{gp} is the commitment key shared between the parent and the grandparent. Λ_{agg} and Λ_p are then sent to the grandparent node. The grandparent adds Λ_{agg} and Λ_p. This addition results in the subtraction of $\sum \gamma_{pc}$ from the shares leaving only γ_{gp}.

1) Corrupt Aggregator Detection: We assume that the attacker can capture a certain percentage of aggregators in the network and all the captured nodes inject false data in the network. With this assumption, various scenarios are possible, under a false data injection attack (See Fig. 2).

The base station detects corrupt data by following the approach in Algorithm 4. When corrupt data is received from a node, the base station queries this node for its shares Λ, the shares Λ_c received from its child nodes, and the commitment keys it shares with the children. As discussed in section VI-D, an intermediate node can change the aggregate and the data encoded in the shares in two ways. If the scrambling key is known, the intermediate node can make changes to the data without making any change to the integrity key. If the scrambling key is not known, both the aggregate and other encoded data, can be changed by making random changes in the shares. Since a node does not share Ψ^2 with any other node, corrupt nodes do not know $\sum \Psi^2$. Thus, they will have to make random changes in the shares. Any random change, however, would cause unpredictable changes in all the enoded data in the shares. When an aggregator receives a query from the base station, it responds with, its own final shares and the shares received from its children, as well as the commitment keys it shares with each of its children. The base station then uses appropriate $\sum \Psi^2$ to extract the commitment key from the shares, verifying it against the one sent by the queried node. If the commitment check fails, the children of this node are queried further for their shares and their children's shares. If the shares sent by the queried node and the ones sent by a child node fail to match, both the nodes are removed from the system, since either the child or the parent is trying to frame the other. If the shares do match and the shares sent by the child node pass the commitment check, the queried node is removed from the network. If the shares sent by the child node fail further, the network is probed further in a similar fashion until a corrupt node is detected.

Security Analysis: An analysis similar to the one performed for basic PIP can be performed for enhanced PIP with same results.

Note: While it may seem that the enhanced PIP algorithm is expensive in terms of communication, we argue that because the detection of corrupt nodes in a network is vital such message complexity can be tolerated. Moreover, the detection phase takes place infrequently and is only needed once corrupt data is detected.

2) Attack scenarios: In a network where multiple nodes are captured by an adversary, captured nodes may try to affect the network in six different ways, as illustrated in Fig. 2. Enhanced PIP algorithm can handle all the six scenarios. In some cases, however, it is impossible to differentiate between a corrupt node and a good one. In such cases, the good node along with the corrupt ones must be removed from the network. In this section we discuss all the six scenarios and how enhanced PIP deals with these cases.

- Case 1. Corrupt nodes only injects false shares
  This case consists of scenarios 2, 3, and 6 in Fig. 2. A

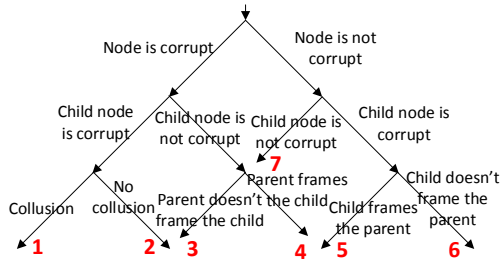

Figure 2. Attack scenarios

standalone corrupt node which only injects false shares will be detected in step 6 of Algorithm 4.

- Case 2. Corrupt child frames a good parent node
  This case is seen in scenario 5. A child node may inject false shares into the system, but when queried for the shares, it submits the original shares, thus framing the parent node. Our algorithm detects this change. It, however, is not possible to determine which node indeed is lying and hence the child node and its parent both will be removed from the network.
- Case 3. Parent node frames the child node
  This case is seen in scenario 4. The parent node changes the shares received from a child node before aggregation, and presents these altered shares when queried. The child node, when queried presents the original shares. Since, for the base station, this case is indistinguishable from Case 2, both the nodes are removed from the system.
- Case 4. Corrupt nodes collude
  This is scenario 1 in Fig. 2. In our algorithm unlike [6], [1], [5], node collusion does not result in any privacy loss. A node can only affect one other node at a time like in cases 1 & 2. Even when the nodes collude there is no more damage than when they are not colluding.

B. Multidimensional PIP

Most motes these days have multiple sensors and in many cases, we need these motes to use more than one of them. This gives rise to multidimensional data. The challenge with multidimensional data is to ensure its privacy and integrity and be energy and bandwidth efficient. A naive solution is to protect the privacy and integrity of each dimension of the data individually. However, because the provenance of all of these dimensions is same more efficient solutions can be achieved. Multidimensional privacy preserving data aggregation was treated in [7]. The authors in [7] did not consider data integrity, we, however, observe that integrity can be built into their solution, in the same manner as ours. Their scheme assumes a heterogeneous network, where the aggregators nodes are few and are more powerful than the rest. This limits the flexibility of their solution as every node must be within the vicinity of a powerful aggregator failing which, it cannot be a part of the data aggregation process.

Both the basic and enhanced versions of PIP can be extended easily to accommodate multidimensional data. Unlike the solution in [7], our solution works in a homogenous environment without the assumption of any powerful aggregator node. We begin with the basic PIP, taking the first dimension as data, adding other dimensions just like the integrity key was added in the basic PIP algorithm. Integrity key is added similarly at the end. For each dimension, this solution adds one share. Thus, the bandwidth usage after the first dimension increases by one share per dimension, compared to the naive solution in which each dimension would require 3 shares.

VIII. Performance Analysis

We implemented the algorithms on the TelosB mote platform using TinyOS 2.x. The TelosB mote platform has a TI MSP430 16 bit microcontroller, with 10 kB of RAM and a clock speed of 8MHz. We measured the execution time of the share generation algorithm and the data regeneration and integrity check algorithm for different data/share sizes and the results are shown in Fig 3. Fig 4 illustrates the corresponding energy consumption for varying data/share sizes. To measure energy consumption we used the formula $E = V * I * t$, where V is the voltage supplied, I is the current drawn, and t is the time taken for the operation. From figures 3 and 4, we can deduce that PIP introduces very little delay in the network and consumes very little energy while providing privacy and integrity of the data. As discussed previously, asymmetric homomorphic encryption and signatures such as in [2] have been used to provide privacy and integrity. Table I shows a comparison between

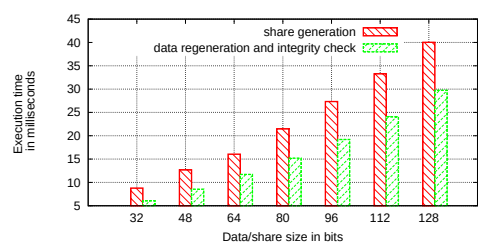

Figure 3. Execution time of Basic PIP on TelosB motes

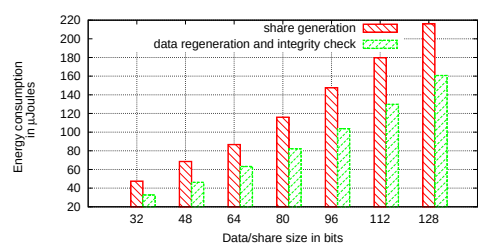

Figure 4. Energy consumption Basic PIP on TelosB motes

Table I
ENERGY EFFICIENCY OF PIP COMPARED TO SDA [2]

Scheme	32/128 bit	48/192 bit
SDA	79.77 mJ	115.7 mJ
PIP	0.047 mJ	0.068 mJ

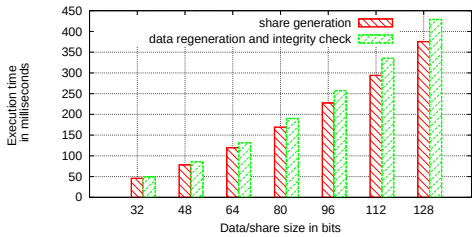

Figure 5. Execution time of enhanced PIP on TelosB motes

the homomorphic SDA algorithm and PIP scheme for two different key sizes. SDA uses Elliptic Curve Cryptography (ECC). According to the 2011 ECRYPT report, a 128-bit ECC key has security equivalent to a 64-bit symmetric key, while the 192-bit ECC key is equivalent to a 96-bit symmetric key. As previously discussed, a b bit share size in PIP has security equivalent to $2b$ bit symmetric key, by virtue of using two b bit keys, the scrambling key and the perturbation key. Therefore, we compare the 128-bit ECC in SDA with 32-bit PIP and 192-bit ECC with 48-bit PIP. It can be seen that, PIP consumes far less energy for equivalent security than SDA. The difference is approximately an order of 4.

We have also implemented enhanced PIP on TelosB motes. Because the calculation of higher degree polynomials is a time-consuming process, we pre-computed several repeatedly used coefficients and placed them in the init function. The execution time for varying data/share sizes in enhanced PIP is shown in Fig. 5. Even after pre-computing the repeatedly used coefficients, we can see that the time required to generate shares in enhanced PIP is much higher than in basic PIP. The energy consumed in enhanced PIP, although not shown, will also be proportionally higher. The memory requirements of basic and enhanced PIP algorithms are shown in Fig. 6. We see that the RAM required by the algorithms is modest, ranging from under 1kB for both basic and enhanced versions for 32-bit share size to a maximum of 1.54 kB for enhanced PIP with a 128-bit share size. Thus both versions of PIP are easy to load on most low memory wireless sensors.

We compare our algorithm with iPDA and iCPDA in terms of bandwidth consumption per node. We consider the header size of the network packet to be 18 bytes, which is the case for TelosB on TinyOS 2.x. For iPDA, to ensure a fair comparison, we assume that the parameter J, which dictates the number of slices of the data item is 3. In iPDA, each node slices its data into J parts and sends out $2J-1$

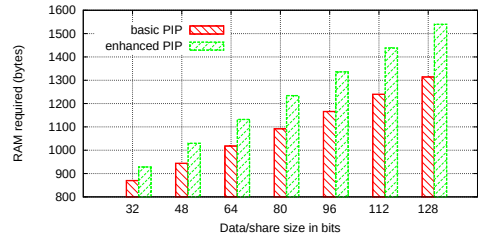

Figure 6. Memory required for basic and enhanced PIP on TelosB.

packets to other nodes in the network. For iCPDA, we take the average size of the cluster $k = 3$. The parameter p_c, which is the number of cluster heads is $1/k$. Thus 1/3 of the nodes in the network are cluster heads. We also assume that the cluster tree of iCPDA has degree 2. In this case then, each node on average sends $(k^2 + k + 1)/2$ packets and receives on average $(3k^2 - 1)/k$ packets. In contrast, in our algorithm, regardless of the number of children each aggregator had, each node sends out 3 shares in a single packet and each aggregator receives d such packets, where d is the degree of the tree taken to be 3 here. The comparison of the bandwidth required per node is illustrated in Fig. 7.

Fig. 8 shows the comparison of energy consumption of the three algorithms. Energy on a node is mainly consumed in the encryption (share generation) operation and in the wireless transmission and reception of the data. For this comparison we vary the number of nodes whose data has to be aggregated. In iPDA, this parameter is J, which governs the number of slices in which a data has to be divided. A node sends the slices to $2J-1$ other nodes. Each slice of the data is encrypted with a shared symmetric key between the pair of nodes. Thus, every node in the network, on average, both sends and receives $2J-1$ encrypted slices of data. This amounts to $2J - 1$ encryptions and decryptions on every node. In iCPDA, this parameter is the cluster size k. Here, each node sends out $k - 1$ encrypted messages to $k - 1$ other nodes in the cluster. All other communication in the cluster is in cleartext. In our algorithm, this parameter may be chosen as the degree of the tree. Our algorithm, however, is agnostic to this metric and hence has a constant computational overhead. The aggregator node however, receives a varying number of packets from its child nodes depending on its degree. This consumes an amount of energy which varies with the number of child nodes each aggregator has. This energy spent in transmission and reception of packets however, is small when compared with iPDA and iCPDA due to lesser amount of wireless communication as shown in Fig. 7. In the experiment, we chose AES-256 encryption for iPDA and iCPDA. In our implementation on TelosB motes, each AES-256 encryption consumes .01 millijoule and decryption consumes .016 millijoule of energy. We assume that the computational overhead in slicing and adding the data in iPDA and iCPDA, and addition of aggregates

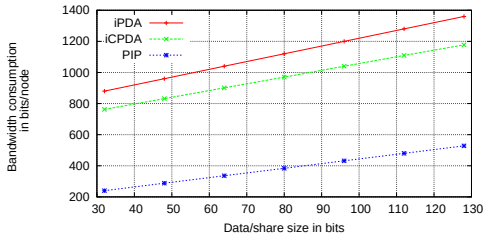

Figure 7. Bandwidth consumption per node for varying share size.

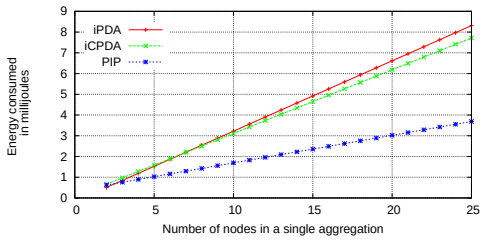

Figure 8. Energy consumed at each node as a function of number of nodes involved in aggregation.

in PIP is negligible. In both iPDA and iCPDA, the privacy of the algorithm depends on the number of nodes taking part in aggregation. The computational overhead, however, increases linearly with the number of nodes and hence a tradeoff must be made between privacy and computational overhead. In our case though, the number of nodes in aggregation does not affect the privacy preserving capability of the algorithm and hence, no such tradeoff is required. As discussed previously, the security of a b bit share in PIP is equivalent a $2b$ bit symmetric key. We thus, choose to compare the two algorithms with the 128 bit version of PIP. In Fig. 8, we can see that PIP performs better in terms of energy consumption when aggregating data for 3 or more nodes in case of iPDA and at all times in case of iCPDA.

IX. Conclusion

In this paper, we first presented the basic PIP algorithm, to perform privacy and integrity preserved data aggregation in wireless sensor networks. We then provided the enhanced PIP algorithm which can detect corrupt aggregators that inject false data into the system. We implemented both algorithms on TelosB motes and measured the execution time and energy consumption. The results establish that the algorithms are lightweight in terms of energy and introduce very little delay in the network. We also compared the basic PIP algorithm's bandwidth requirement with two other existing algorithms with similar objectives and found that basic PIP performs better than both. Finally, we noted that while the privacy in other algorithms depended on the number of nodes taking part in aggregation, in our algorithms it is not so and this also helps us save energy at the aggregators.

References

[1] W. He, X. Liu, H. V. Nguyen, K. Nahrstedt, and T. Abdelzaher, "Pda: Privacy-preserving data aggregation for information collection," *ACM Trans. Sen. Netw.*, vol. 8, no. 1, pp. 6:1–6:22, Aug. 2011.

[2] V. Kumar and S. K. Madria, "Secure hierarchical data aggregation in wireless sensor networks: Performance evaluation and analysis," *Mobile Data Management, IEEE International Conference on*, vol. 0, pp. 196–201, 2012.

[3] J. Girao, D. Westhoff, and M. Schneider, "Cda: concealed data aggregation for reverse multicast traffic in wireless sensor networks," in *Communications, IEEE International Conference on*, vol. 5, May 2005, pp. 3044 – 3049 Vol. 5.

[4] T. Feng, C. Wang, W. Zhang, and L. Ruan, "Confidentiality protection for distributed sensor data aggregation," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, pp. 56–60.

[5] L. Zhang, H. Zhang, M. Conti, R. Pietro, S. Jajodia, and L. Mancini, "Preserving privacy against external and internal threats in wsn data aggregation," *Telecommunication Systems*, pp. 1–14, 2011.

[6] M. Conti, L. Zhang, S. Roy, R. Di Pietro, S. Jajodia, and L. V. Mancini, "Privacy-preserving robust data aggregation in wireless sensor networks," *Security and Communication Networks*, vol. 2, no. 2, pp. 195–213, 2009.

[7] X. Lin, R. Lu, and X. S. Shen, "Mdpa: multidimensional privacy-preserving aggregation scheme for wireless sensor networks," *Wirel. Commun. Mob. Comput.*, vol. 10, no. 6, pp. 843–856, Jun. 2010.

[8] C. Wang, G. Wang, W. Zhang, and T. Feng, "Reconciling privacy preservation and intrusion detection in sensory data aggregation," in *INFOCOM, 2011 Proceedings IEEE*, 2011.

[9] W. He, H. Nguyen, X. Liu, K. Nahrstedt, and T. Abdelzaher, "ipda: An integrity-protecting private data aggregation scheme for wireless sensor networks," in *MILCOM 2008. IEEE*, nov. 2008, pp. 1 –7.

[10] W. He, X. Liu, H. Nguyen, and K. Nahrstedt, "A cluster-based protocol to enforce integrity and preserve privacy in data aggregation," in *Distributed Computing Systems Workshops, 2009. ICDCS Workshops '09. 29th IEEE International Conference on*, June 2009, pp. 14 –19.

[11] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[12] A. Parakh and S. Kak, "Recursive secret sharing for distributed storage and information hiding," ser. ANTS'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 88–90.

[13] C. Castelluccia, A. C.-F. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 5, no. 3, pp. 20:1–20:36, Jun. 2009.

View publication stats