Confidentiality-Preserving and Fault-Tolerant In-Network Aggregation for Collaborative WSNs

Marian K. Iskander, Adam J. Lee, Daniel Mossé
Department of Computer Science, University of Pittsburgh
{marianky, adamlee, mosse}@cs.pitt.edu

***Abstract*—In Collaborative WSNs, sensing devices are owned and operated by different stakeholders with incentive to preserve the confidentiality of their individual sensors readings while contributing to statistics computed by the group. In such systems, in-network data processing presents high efficiency for energy and bandwidth, but unfortunately introduces several challenges related to data security and fault tolerance.**

In this paper, we present and analyze a new protocol that allows for confidentiality-preserving in-network aggregation for collaborative WSNs in the face of intermittent link-level failures. Our protocol makes use of a symmetric-key, homomorphic cryptosystem to maintain the confidentiality of individual sensor readings while still permitting a trusted sink node to recover the correct aggregate value. The encrypted sensor readings are then combined using a multipath aggregation protocol that is capable of detecting and recovering from a variety of failure scenarios while carefully handling duplicate sensitive aggregates. We prove the security and correctness of our protocol, and we conduct simulation studies to understand its data transmission and energy consumption overheads.

Our results show that both confidentiality and fault tolerance can be achieved in representative network configurations, while incurring an increase of 7.1% in the average message size and 3.6% in the average energy consumption. In the unlikely scenario that 100% of the sensor nodes participate in an aggregate query, the average energy consumption showed at most a 25% increase.

***Keywords*-Collaborative wireless sensor networks, fault-tolerance, in-network aggregation, confidentiality, homomorphic encryption**

I. Introduction

A wireless Sensor Network (WSN) is a collection of tiny devices capable of sensing, computing, and wirelessly communicating to monitor and control events of interest in a distributed manner. WSN applications span various domains such as environmental and building monitoring and surveillance, pollution monitoring, agriculture, health care, and energy management. Through collaboration, wireless sensors owned and operated by different entities and stakeholders can be used to collaboratively sense and detect phenomena of shared interest.

In *Collaborative WSNs (CWSNs)*, in-network data aggregation has been adopted as an energy-efficient process that allows each node along the routing path to aggregate *all* values received from its children into a *single* response value (thereby avoiding the transmission of messages from each sensor to the data sink). However, different schemes for in-network aggregation impose different challenges. *Tree-based* in-network aggregation provides the minimal communication overhead by using a spanning tree across all sensor nodes, but a single link failure in this model leads to the loss of all data from the subtree connected by that link. Given that WSNs are characterized by high rates of communication failures (up to 30% loss rate [1]), this approach can lead to large errors in the average case. *Multipath-based* in-network aggregation approaches add robustness to the traditional tree structure by taking advantage of the broadcast medium, but must be carried out carefully to avoid overcounting when computing aggregates.

In addition to achieving the goal of reliably executing the aggregation process, CWSNs require that the confidentiality of individual sensor readings be preserved. Without such a guarantee, different sensors' stakeholders could gain useful information from the confidential sensor readings and adversaries within the proximity of the network could infiltrate the network, eavesdrop, and gain useful information as well. Unfortunately, existing mechanisms for carrying out confidential in-network aggregation either require the use of expensive cryptographic primitives that are unsuitable for use in resource-limited sensor environments (e.g., [2]), or assume perfectly reliable communication links (e.g., [3], [4]).

Motivating Application: One example of *Collaborative Sensing over Shared Infrastructure (CSSI)* applications is for office buildings that are equipped with a sensing infrastructure that is shared among many stakeholders. In some situations, different stakeholders may not want to share information about the occupancy of individual rooms, although they might want to contribute in computing statistics about the occupancy of regions within the building in order to make better decisions about heating/cooling, public safety, facilities surveillance, and traffic monitoring. Sensors managed by individuals or departments within the building could measure statistics like occupancy or temperature, encrypt their results, and forward the encrypted results through the shared sensing infrastructure. These readings are then aggregated *in encrypted form*, thereby reducing overheads in the network while protecting the individual values sensed. In the end, the aggregate value(s) are decrypted to derive the desired statistics. Note that, in this case, the preservation of each value's confidentiality provides user location privacy by not disclosing, say, the occupancy status of individual offices. ♦

In this paper, we present a protocol for reliably carrying out in-network aggregation in CWSNs exhibiting link failures while also maintaining the end-to-end confidentiality of indi-

vidual sensor readings. To achieve secure data aggregation we have chosen to use end-to-end encryption of sensor readings over hop-by-hop encryption for the following reasons:

- Using end-to-end encryption, intermediate nodes cannot decrypt the readings that they forward and thus data are not only protected from external eavesdroppers, but also from malicious compromised or curious nodes within the collaborative WSN.
- To aggregate data encrypted in a hop-by-hop manner, each sensor node must first decrypt each encrypted value that it has received, aggregate the resulting values, and then re-encrypt the aggregate. Hence, the overhead of the decrypt-aggregate-reencrypt operations of hop-by-hop increases linearly in the number of children of each node.
- End-to-end encryption eases the key distribution process by reducing the number of keys required by the system (a sensor shares only one unique secret key with the sink node).

Our protocol uses a lightweight homomorphic cryptosystem [5] to enable collaborative in-network aggregation of encrypted values while imposing small computational overhead on individual sensors. This aggregation takes place by extending the RideSharing multipath aggregation protocol [6] to maintain additional metadata that allows the sink node to recover the key needed to decrypt the hidden aggregate value. The scheme presented in this paper was designed with the following goals in mind: **(a) Confidentiality**: individual sensor readings and their aggregate values are not revealed to any external (eavesdropper) or internal (compromised aggregator) attacker, only to the sink; **(b) Fault tolerance**: robustness in that sensor readings that are lost due to link errors are compensated for *at most once*; **(c) Exact aggregation**: Instead of providing probabilistic query aggregate results (e.g., [7], [8]), our approach provides an exact aggregate result in case of no link failures. With link failures introduced, the final aggregate deviates by exactly the lost value and not by some derivative of that value; and **(d) Low energy overhead**: our scheme has low overheads on the size of packets transmitted and amount of computation required.

The rest of the paper is organized as follows. Section II reviews the different in-network data aggregation schemes proposed in the literature. Section III presents our network and attack model, as well as the basic building blocks for our new protocol. In Section IV, we present our new proposed confidentiality-preserving, fault-tolerant, in-network aggregation protocol, and discuss its relevant properties. Section V presents an extensive simulation study and system evaluation of our approach. Finally, Section VI presents our conclusions and directions for future work.

II. Background and Related Work

There has been extensive work on data aggregation schemes including (e.g., [9], [10], [11]). These schemes assume both a reliable and a secure network. However, in the real world, sensor nodes are usually deployed in hostile environments where communication links can be an easy target for adversarial eavesdropping. The reliability assumption is also a strong one, as sensor networks are subject to frequent link failures that might cause the loss of a whole sub-tree aggregate value. Below we describe works that address fault-tolerant or confidentiality-preserving in-network aggregation.

a) Confidentiality Preserving Systems: A simple approach to partially preserving the confidentiality of aggregated data in WSNs is using hop-by-hop encryption ([12], [13]). The problem with all hop-by-hop encryption schemes is that, if the node has been compromised, they may violate the confidentiality of the data at each node since the data is decrypted before the aggregation. Besides the confidentiality violation at nodes, the number of necessary shared secret keys becomes a function of the network density where all neighboring sensor nodes must share secret keys.

A higher level of confidentiality is achieved through complete end-to-end encryption of the sensor readings and their aggregate values. The use of homomorphic cryptosystems such as RSA, ElGamal, Elliptic Curve [14], or Paillier [15], would allow for end-to-end encryption with in-network aggregation. Unfortunately most such algorithms require extensive computations and very long keys that do not suit the computational and power constraints of WSNs. To enable additive aggregations of sensor values, our approach makes use of the additively homomorphic symmetric-key stream cipher proposed by Castelluccia et al. in [5], which is a simple and efficient cipher system and well-suited for use in WSNs.

Existing confidentiality-preserving aggregation protocols (e.g., [3], [4]) share one common assumption: no packets are lost during the aggregation process (note that link failures are common due to the unreliable communication medium of WSNs). In fact, in some proposed schemes such as [16], additive aggregation is supported using a secret splitting technique for the sensor readings. In such scheme, a single packet loss would cause the whole aggregation process to fail and the inability of the sink node to recover a precise aggregation result. The authors in [3] proposed an end-to-end encryption for sensor networks traffic, but their scheme is not resilient against link failures. Their proposed scheme utilizes encrypted default values to compensate for link failures, which does not solve the problem of delivering the correct values when link failures happen. By contrast, our approach for confidentiality-preserving in-network aggregation is robust against individual link failures. That is, rather than assuming a silent node is non-operational, our approach allows the parents of silent nodes to defer the aggregation of these nodes' values to other *backup* parents that may have overheard the perceived silent node's transmission. This allows the computation of more accurate aggregate values.

b) Fault Tolerance Models: In typical in-network data aggregation systems, a query disseminates from a sink node to all other sensor nodes. As the query propagates in the network, a spanning tree is constructed. The spanning tree is rooted at the sink node, and each intermediate sensor node receives values from its children, aggregates them with its own value, and forwards the result to its parent. One major drawback of

spanning trees is that they are not robust against link failures: a single link failure causes the value of the sub-tree connected by this link to be lost. Link failures that occur in the upper levels of a tree can thus cause massive losses of data. To address this problem, existing mechanisms (e.g., [17], [18]) make use of multipath in-network aggregation. This approach adds more robustness against link failures, but at the same time introduces challenges with correctly handling duplicate-sensitive aggregates such as SUM, AVG, and COUNT.

Different variations of so-called *sketches* provide approximate aggregation in WSNs for duplicate insensitive queries (e.g., [8] are based on FM-sketches [19]). Unfortunately, these solutions are not applicable because aggregate computations are infeasible when using encrypted sketches.

In [6], RideSharing is introduced as a fault-tolerant scheme for duplicate sensitive aggregations in WSNs. RideSharing has been shown to outperform other fault tolerant schemes (e.g., Synopsis Diffusion [18]), consuming up to 50% less energy and bandwidth resources, while delivering more accurate aggregate results.

No published work so far solves the problem of providing both confidentiality and fault tolerance for the process of in-network aggregation. Hence, our approach is novel in that it accomplishes *both* of these goals in an energy efficient manner for collaborative WSNs.

III. Models and Building Blocks

A. Network and Attack Model

We assume a multi-hop network that consists of n static sensor nodes and a single trusted sink node. Each sensor node shares a unique symmetric key with the sink. As usual, sensors are small, battery-operated devices and the sink is a more capable node with higher computational and storage capabilities and no battery limitations. The sensors may belong to different stakeholders and execute collaborative sensing applications that use in-network aggregation to efficiently compute statistics over the individual sensor readings. We are concerned with the data confidentiality, no bound on the number of attackers, and consider two types of attackers: (a) *honest but curious* sensors, and (b) *quiet infiltrators*. Both are able to eavesdrop and either accumulate the information gathered or send the information in an undetected way (e.g., using a different channel). The sink node is assumed to remain uncompromised. Faults are in links only and they are "omission" faults or "crash" faults.

B. Cryptographic Primitives

Our confidentiality preserving scheme makes use of the symmetric key, additively homomorphic stream cipher proposed in [5]. In this cryptosystem, a keyed pseudo-random generator is used to effectively generate keystreams that are used to encipher sensor readings stored as integer values. Encryption is simply addition mod M and decryption is subtraction mod M, where M is an upper-bound on the aggregate function to be computed. For example, a sensor node sharing a key k with the sink and using pseudo-random generator g can encrypt its j^{th} reading of a value, v^j, as follows:

$$c^j = v^j + g^j(k) \bmod M$$

The sink can then recover the value v^j as follows:

$$v^j = c^j - g^j(k) \bmod M$$

A key feature of this cryptosystem is its ability to homomorphically combine values that are encrypted under the same or different keys. Consider two sensor nodes n_1 and n_2 sharing keys k_1 and k_2, respectively, with the sink (but not with each other). Suppose these principals wish to encrypt their i^{th} values v_1^i and v_2^i, respectively. The nodes encrypt their values as follows:

$$c_1^i = v_1^i + g_1^i(k_1) \bmod M \quad (1)$$
$$c_2^i = v_2^i + g_2^i(k_2) \bmod M \quad (2)$$

Given the aggregate value $C^i = c_1^i + c_2^i$, the sink can recover the aggregate key $K^i = g_1^i(k_1) + g_2^i(k_2)$ and decrypt the aggregate value $V^i = (v_1^i + v_2^i) = C^i - K^i \bmod M$. Note that neither v_1^i or v_2^i are disclosed via this process.

In [5], the authors prove that this symmetric key, additively homomorphic cipher is semantically secure.

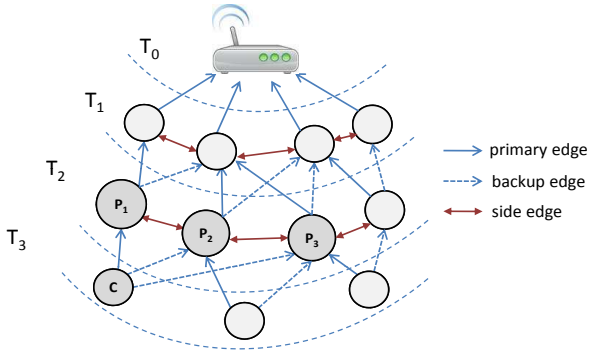


Fig. 1. Track graph network topology

C. Cascaded RideSharing

Cascaded RideSharing was proposed in [6], as an efficient fault tolerant in-network aggregation. Cascaded RideSharing exploits the redundancy in the wireless medium to detect and correct communication link failures. To accomplish this, the sensor network is organized into a *track graph* topology [20]. Figure 1 illustrates a simple track graph topology. In such a topology, sensor nodes are organized in tracks, with the sink residing in track 0, sensors one hop away from the sink are in track 1, and so forth. The aggregation path then forms a DAG with multiple paths through the track graph, rather than a simple spanning tree. Each sensor node has one *primary* parent and one or more *backup* parents in the adjacent track (a track consists of all nodes at the same distance from the sink).

A parent of a sensor node is assumed to overhear another parent's transmissions. Note that the assumption of overhearing sensors (which has also been adopted in other work ([21], [17]) does not constrain the WSN deployment to only dense networks because such assumption is easily justified when a

sensor node is within a reachable distance of at least two parents. Every sensor node transmits its reading to its primary parent according to a predefined TDMA schedule. Note that not all sensors transmit data, only those that include new information, as determined by the application. For example, repeatedly transmitting the same temperature or other information would not add any new value, and therefore the application may choose to omit redundant/repeated values [22].

Backup parents compensate for errors in the primary links by overhearing another parent's transmissions. For error detection and correction purposes, each parent maintains a small bit vector L that has two bits for each child: r-bit (retransmitted bit) and e-bit (error bit). If the primary parent does not receive any data from a specific child node in its predetermined time slot, it sets the e-bit to '1' indicating a missing value from this child. A backup parent then takes the responsibility of aggregating the missing value. The r-bit is set to '1' if the parent is able to aggregate and transmit the value for the child node. This process repeats among backup parents until one parent aggregates the value. Note that the primary parent of some node might also function as a backup parent for other nodes.

IV. Confidentiality Preserving and Fault Tolerance Protocol

In this section, we describe our protocol for confidentiality-preserving and fault-tolerant in-network aggregation among CWSNs. We begin by discussing the details of our protocol, and then prove that it affords strong reliability and confidentiality properties.

A. Protocol Overview

At a high level, our approach to providing fault-tolerant and confidentiality-preserving in-network aggregation for CWSNs is built on top of the Cascaded RideSharing algorithm combined with the additively-homomorphic stream cipher described in Section III-B. In the event that the readings of *all* sensor nodes are included in the final aggregate value, the algorithm goes as follows: (i) each sensor n_i encrypts its value v_i as $c_i = v_i + g_i(k_i) \bmod M$; (ii) the resulting c_i values are aggregated using the Cascaded RideSharing protocol, which results in the sink receiving the value $C = \sum_i c_i \bmod M$; (iii) the sink then computes the aggregate key value $K = \sum_i g_i(k_i) \bmod M$; and (iv) the sink extracts the final aggregate value $V = \sum_i v_i = C - K \bmod M$.

Unfortunately, the above algorithm only works in the rare case that *all* sensor nodes contribute readings to an aggregate computation. Most commonly, not all sensors' readings are included in the final aggregate because of either node- or link-level failures or simply because not every sensor will have a reading to contribute to every query. In this case, the sink node would compute an incorrect aggregate key K. If the sink attempts to decrypt the aggregate ciphertext using the wrong aggregate key, the resulting value will be a random element from the set $\{0, \ldots, M-1\}$. This random and unbounded error is due to the semantic security of the cipher, which ensures that a ciphertext reveals no information about the corresponding plaintext without the appropriate key.

To account for the above types of problems, we designed our protocol in such a way to allow the sink node to *efficiently* determine which sensors contributed readings to the final aggregate and thus correctly compute the aggregate key that should be used to recover the true aggregate value from the ciphertext received. As we will see, this is achieved by propagating state not only between nodes within the same track as done in RideSharing, but also between nodes in adjacent tracks. Using our protocol, even duplicate-sensitive aggregates such as SUM and AVG are securely computed using the in-network aggregation process, while guaranteeing with high probability that every sensor reading contributes to the final aggregate at most once.

B. Protocol Details

Algorithm 1 contains pseudo-code describing the aggregation protocol as run by sensor nodes that help aggregate and route readings in the network, and optionally contribute their own readings to the aggregate being computed. This algorithm takes four inputs: a set of child nodes for which this node is the primary parent ("Primary" Children, or PC), a set of child nodes for which this node is a backup parent ("Backup" Children, or BC), the list of peer nodes in this track (set of peers, or SP), and an optional sensor reading to include in the aggregation (value v). In addition to maintaining the fault-tolerant L bit vector needed by the Cascaded RideSharing protocol (cf. Section III-C), Algorithm 1 also maintains a *Partaking vector*, called the P vector, to keep track of nodes that have successfully contributed to the final aggregate. The P vector is an n-bit vector, where n is the number of sensor nodes in the network. In Section V, we show that in practice, sensor nodes do not necessarily have to transmit the whole P vector, but only a compressed vector to minimize bandwidth overhead.

The protocol proceeds as follows. If the sensor node has a non-null reading v to contribute to the aggregate, v is first encrypted using the homomorphic cryptosystem described in Section III-B and then added to the local aggregate A. After receiving P and L vectors, the node sets the bit corresponding to its ID in the P vector to '1', indicating that it has contributed to the aggregate value. The sensor then waits to receive the L vectors transmitted by the nodes in its track that precede it in the TDMA transmission order; the bit vector informs the sensor of what corrective actions it should attempt to take. After receiving P and L vectors, the sensor iterates over all of its child nodes and combines the aggregate values and P vectors reported by these nodes with its local values as indicated by the L vector. Specifically, values are included into the sensor's local aggregate if this sensor is the primary parent of the child, or if it is a backup parent that is capable of correcting an error induced by faults that affect the child's transmitted value.

After receiving data from all its child nodes, the sensor transmits its updated aggregate value A, its updated P vector, and its local L vector. This message is destined to its parent

Algorithm 1: Aggregation and routing algorithm run by sensors within the network

```
input : PC, BC, SP, v
A := 0;
P := 0̄;
L.r := 0̄;
L.e := 0̄;
if v NOT NULL then // Aggregate own value
    A := A + v + g_ID(k_ID) mod M;
    P[ID] := 1;
end
L := rcvL(SP);
foreach Child C in PC ∪ BC do
    if rcv(A_c,P_c) from Child C then
        if C ∈ PC OR (C ∈ BC AND L[C].e = 1
        AND L[C].r = 0) then    // Aggregate
        the received values
            A := A + A_C mod M;
            P := P OR P_c;
            L[C].e := 1;
        end
    end
    else   // Propagate the error signal
        L[C].e := 1;
    end
end
Transmit(A,P,L);
```

Algorithm 2: Final aggregation and decryption algorithm used by the data sink

```
input : PC
output: FinalA
A := 0;
P := 0̄;
K := 0;
FinalA := 0;
foreach Child C in PC do
    if rcv(A_c,P_c) from Child C then
        A := A + A_C mod M;
        P := P OR P_c;
    end
end
foreach bit set to '1' in P do
    K := K + g_i(k_i) mod M;
end
FinalA := A − K mod M;
```

nodes (primary and backup) and to the peer parents (backup for its children).

Note that the L vector is a local vector used merely for the coordination of primary and backup parents through side edges within each track; that is, the L vector is re-created in each track and ignored by nodes in adjacent tracks. On the other hand, the P vector propagates to the next track along with the aggregate value.

Algorithm 2 contains pseudo-code describing the protocol run by the sink node requesting the aggregate. This algorithm takes only a single input: the set of children in track 1 of the graph (PC, similar to Algorithm 1). After the sink receives an encrypted value and a P vector from each of its responsive children, it computes the sum of each such A value and the bitwise OR of every P vector to compute both the final (encrypted) aggregate value and the final P vector indicating which nodes successfully contributed to the aggregate. The sink then generates the keystreams for each node indicated in the final P vector and uses the aggregate key to recover the plaintext aggregate value.

C. Protocol Properties

We now show that our protocol provides strong guarantees in terms of both sensor reading confidentiality and correctness of the final aggregation.

Theorem 1 (Confidentiality): During the execution of the protocol described by Algorithms 1 and 2, no sensor (except the sink) can learn the value of the readings reported by any other sensor, nor the value of any intermediate aggregate value.

Theorem 1 follows directly from the semantic security of the cipher used by Algorithms 1 and 2 and the fact that each sensor node shares a unique symmetric key only with the sink.

Theorem 2 (Correctness): Under the assumption of "honest but curious" or "quiet infiltrators" attack nodes, the protocol described by Algorithms 1 and 2 includes each sensor reading at most one time during the aggregation process. Further, the sink node is able to correctly identify the sensors that contributed to this aggregate, generate the resulting aggregate key, and recover the correct result.

Proof: To prove this claim, we must show that (i) each sensor reading is aggregated at most once and (ii) that the P vector includes exactly the information needed to recover the aggregate key needed to decrypt the result. Note that Algorithm 1 sets a bit in the P vector if and only if the sensor reading for the corresponding node is included in the local aggregate. Also, the P vector is always transmitted with the aggregate values. As such, proving assertion (i) is sufficient to prove the theorem. We proceed by induction on the height of the track graph.

For our base case, we consider a track graph consisting of three tracks: the sink (track 0) and two tracks of sensors (tracks 1 and 2). Assuming that track 1 has perfect connectivity to the sink, we need only show that all readings from track 2 are aggregated by at most one node in track 1. Without loss of generality, we assume that track 2 consists of a single sensor node n_i and consider 5 cases: (1) there are no link failures in the graph, (2) the link between n_i and its primary parent fails, (3) the link between n_i and its primary parent and some number of its backup, (4) there is a side-channel error in track 1 *prior to* the aggregation of n_i's reading, and (5) there is a side-channel error in track 1 *after* the aggregation of n_i's value.

1) If no links fail, the reading of sensor n_i will be heard by its primary parent. The primary parent will include

this value in its local aggregate, set the r-bit for node n_i in its L vector, and transmit. Since the r-bit for node n_i is set, no backup parent in track 1 will take corrective action to include n_i's reading in its local aggregate.

2) If the link between n_i and its primary parent fails, the primary parent will set the e-bit for n_i in its L vector and transmit this vector along its side-channel to the other nodes in track 1. The first backup parent of n_i will then incorporate n_i's reading into its local aggregate, set the r-bit for n_i in its L vector and transmit. No other backup parent in track 1 will take corrective action to include n_i's reading in its local aggregate since n_i's r-bit is now set in the L vector passed along track 1.
3) If the links between n_i and its primary parent and between n_i and some number of its backup parents fail, n_i's primary parent will set the e-bit for n_i in its L vector to indicate an error. This L vector will propagate along the side-channel in track 1 until it reaches a backup parent that has overheard n_i's transmission. If no such backup parent exists, n_i's reading is lost. Otherwise, the first such backup parent incorporates n_i's reading into its local aggregate, sets the r-bit for n_i in its L vector and transmits. No other backup parent in track 1 will take corrective action to include n_i's reading in its local aggregate since n_i's r-bit is now set.
4) If there is a side-channel error in track 1 *prior to* the incorporation n_i's reading in the aggregate, no entry will exist in the L vector for node n_i. As such, nodes optimistically assume that n_i's reading was already aggregated, and will not incorporate n_i's reading. This implies that n_i's reading will be absent from the final aggregate.
5) If there is a side-channel error in track 1 *after* the incorporation of n_i's reading in the aggregate, this implies, as in the last case, that no entry will exist in the L vector for node n_i. As such, nodes optimistically (and correctly) assume that n_i's reading was already aggregated and will not again incorporate n_i's reading.

The above cases account for all possible link failure scenarios between tracks 2 and 1, and within track 1, and in all cases n_i's reading was included at most once. Thus, we have shown that Theorem 2 holds in the base case.

For the induction step, assume that Theorem 2 holds for all track graphs containing up to k tracks. We now prove that it also holds for all track graphs of up to $k+1$ tracks.

First, observe that an argument similar to that used in the base case shows that the reading reported by each sensor in track $k+1$ will be incorporated by at most one sensor in track k. Furthermore, our inductive hypothesis can be used to prove that the value reported by sensor in track k is incorporated at most once into the final aggregate. As such, the readings of sensors in track $k+1$ are incorporated at most once into the final aggregate, and Theorem 2 holds in all track graphs. ■

Taken together, these theorems show that the protocol described by Algorithms 1 and 2 does indeed provide confidentiality-preserving and fault-tolerant in-network aggregation functionality for wireless sensor networks.

D. Integrity Checking

Integrity checking of the data could be another desirable property of CWSNs to defend against external attackers who could manipulate the data in transit. Even though integrity protection is not within the scope of the paper—given our eavesdropping attack model—for completeness, we discuss in this section how to add integrity to our proposed protocol.

One major problem with homomorphic cryptosystems is that they are *malleable* by design. An encryption algorithm is malleable if it is possible for an adversary, without knowing the secret key, to transform a ciphertext into another ciphertext that decrypts to a *related* plaintext. Consider the cryptographic primitives in Section III-B, where nodes n_1 and n_2 encrypt their i^{th} values v_1^i and v_2^i into the two ciphertexts c_1^i and c_2^i and the aggregate value $C^i = c_1^i + c_2^i$. An external attacker can jam the network while the aggregate C^i is being transmitted and then transmit an inflated (or deflated) aggregate value $C^{i'}$, by adding (or subtracting) some constant to C^i *without* having the ability to decrypt the aggregate C^i. When the sink node attempts to decrypt the aggregate it receives $C^{i'}$ it will recover the modified value $V^{i'}$ rather than the true aggregate V^i. Unfortunately, such attacks are undetectable without adding extra cryptographic mechanisms to verify both the data integrity and the authenticity of the encrypted aggregate values across sensor nodes.

It is straightforward to extend our protocol with hop-by-hop integrity protections to guard against the injection or modification of data by malicious outsiders. In order to accomplish this, every sensor node can establish a shared secret key with all its parents in the adjacent track. Secure key distribution between nodes is a well-explored problem, and could be achieved using any secure and efficient key distribution scheme in the literature (e.g., [23], [24]). Using this shared key, the sensor node can compute cryptographic integrity code (e.g., a keyed HMAC [25]) over its aggregate value, and then transmit the aggregate along with its corresponding integrity code to its parents. The parent receiving these values can then verify both the integrity of the message, as well as authenticate that it was sent by one of its legitimate children. If the verification passes, the values are processed as usual. If the verification fails, the faulty value can be ignored by the receiving parent and, if necessary an alert can be raised indicating that tampering has been detected.

Note that in the case of malicious compromised aggregating nodes, integrity checking becomes a more complex problem as end-to-end integrity checking will be required. We leave the solution for this problem to our future work.

V. Evaluation

We carry out a detailed evaluation of the communication and energy-consumption overheads associated with our protocol.

A. Simulation Setup

To understand the costs and benefits of our approach, we implemented four protocols by extending the WSN in-network aggregation simulator TiNA developed in [22]. Specifically,we

implemented (i) a spanning-tree aggregation without fault-tolerance nor data confidentiality; (ii) the Cascaded RideSharing protocol [6], which provides only fault tolerance; (iii) the basic version of our protocol described in Section IV, with both fault-tolerance and data confidentiality protection; and (iv) an enhanced version of our protocol that applies compression (run-length encoding or RLE) to the P vector to minimize data transmission overheads.

All protocols were compared relative to three main metrics:

- *Average relative RMS error*: The root mean square error of the final result, normalized to the correct result value.
- *Average energy consumed per node per epoch*: The average energy spent transmitting, listening for, and receiving data by each node for an epoch.
- *Average message size transmitted per node per epoch*: The average amount of data transmitted by each node during one run of the protocol.

Sensor nodes are distributed over a 320×320 ft^2 grid, with the data sink located closest to the center of this area. The radio range of each node is assumed to be 30 ft. All results are the averages over 10 simulation runs, each with 30 epochs. As in [6], we assume that sensor nodes have the Mica2 specifications [26] where data transmission, listening and reception, and idling consume 65 mW, 21 mW, and 0 mW, respectively; network bandwidth is assumed to be 38.4 $Kbps$. We also assume network activity is much more costly than computation [27].

TABLE I
SIMULATION PARAMETERS

Parameter	Values
Total number of nodes	300, 400, 500, …,1000, 1024
Link error rate	0.05, 0.10, …, 0.35
Number of primary+backup parents	at most 3
Participation level (% of nodes reporting values)	1.5%, 2.5%, 5%, …, 25%

We assume that link errors occur independently of each other and are distributed uniformly throughout the network. Failures can happen independently in primary, backup, and side edges, while the links between track 1 and the data sink are assumed to be error free. The number of (primary and backup) parents for each node is set to be at most 3 and depends on the network (in particular, it is a function of network density). Table I lists some of the simulation parameters.

We implemented the RC4 stream cipher as our pseudo-random keystream generator, due to its simplicity. We also used an appropriate optimization to conserve both energy and bandwidth. Given the nature of the P vector, we applied Run Length Encoding (RLE) to compress the P vector at each sensor node. Using this performance optimization, each sensor node contributing a value sets the corresponding bit in the P vector, aggregates any received P vectors from its children into a single P vector, compresses the new vector using RLE, and sends the compressed version.

With respect to the energy consumed during computation (including aggregation and RLE compression), we rely on the fact that transmitting one bit over radio is at least three orders of magnitude more expensive in terms of energy consumption than executing a single instruction [27]. Our measurements

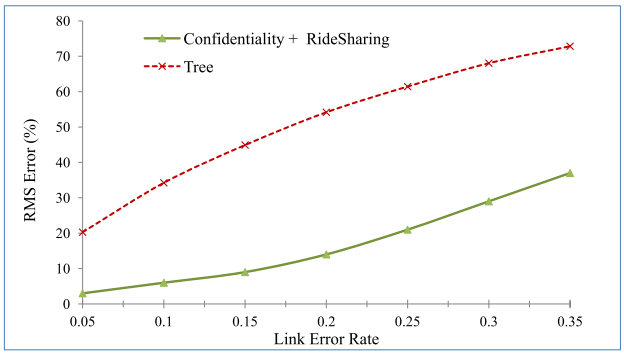


Fig. 2. Average Relative Mean Square Error for 100% participation (parents/node = at most 3, total nodes = 1024)

determined that the number of CPU cycles necessary to compress a P vector of size 1024 bits is less than 20 cycles per bit on the average. Since this cost is greatly dominated by the cost of transmitting a single bit, we do not consider the energy consumption due to compression in our simulations computations. The cost of the simple operations—such as additions, ANDs, ORs, and swaps—required by our stream cipher is similarly dominated by transmission cost, particularly in the event that sensors are pre-keyed and need not generate their keystream on the fly.

B. Experiments

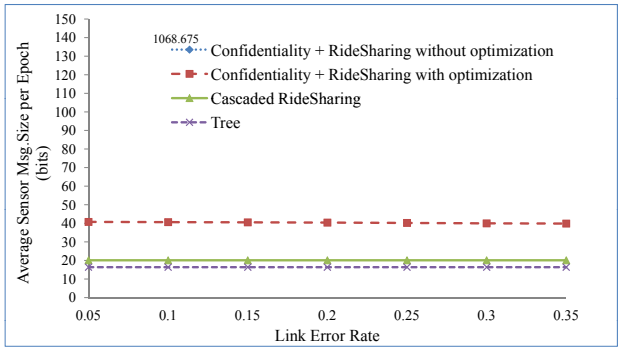
All results below have a maximum error of less than 1% when computed with 95% confidence level. We omit error bars from the plots for better presentation.

1) Effect of Link Error Rate: In this experiment we fix the number of sensor nodes in the network to 1024 nodes, deployed in a grid. The participation level is set at 100%; that is, all sensor nodes participate by contributing readings to the aggregate computation. We vary the link error rates, where the link error rate is the probability with which a link will fail during data transmission.

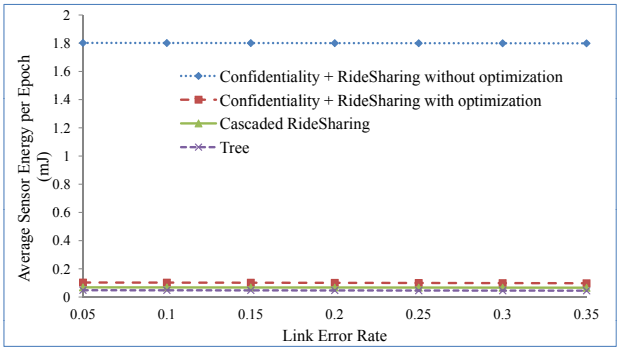
Figure 2 shows the effectiveness of the confidentiality preserving RideSharing and the spanning tree schemes with respect to the RMS error of the aggregated value for different link error rates. As expected, link error is highly correlated with spanning tree error. For link error rate of 35%, we see an improvement in the RMS error of the confidentiality preserving RideSharing scheme over the spanning tree scheme by 48.2%.

Figures 3a and 3b illustrate the average message size and the average energy consumption overheads, respectively, for each scheme. The four schemes each show a stable overhead for different link error rates. Yet, the overhead differs significantly from one scheme to another. The naive version of our new scheme shows the maximum overhead[1] in both message size and the energy consumption. On the other hand, the version with compression optimization shows a significant improvement in the overhead over that without the compression optimization and is comparable to the schemes

[1]It is 50x bigger and thus not shown in Figure

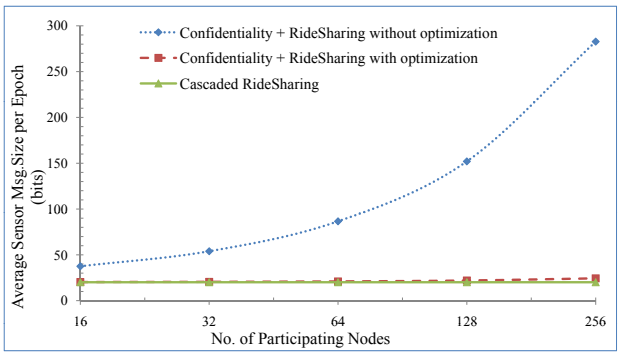


(a) Average message size per node; note that adding the uncompressed P vector causes the average message size to go to 1068 bits (about 50x more than the baseline), and is not shown

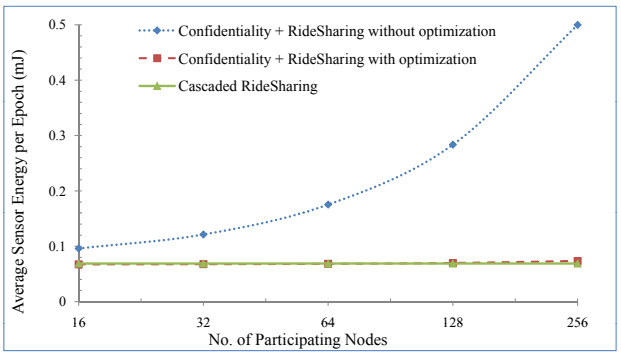


(b) Average energy consumption per node

Fig. 3. Avg. msg. size and avg. energy consumption per node (per epoch) for 100% node participation (parents/node = at most 3, total nodes = 1024)



(a) Average message size per node



(b) Average energy consumption per node

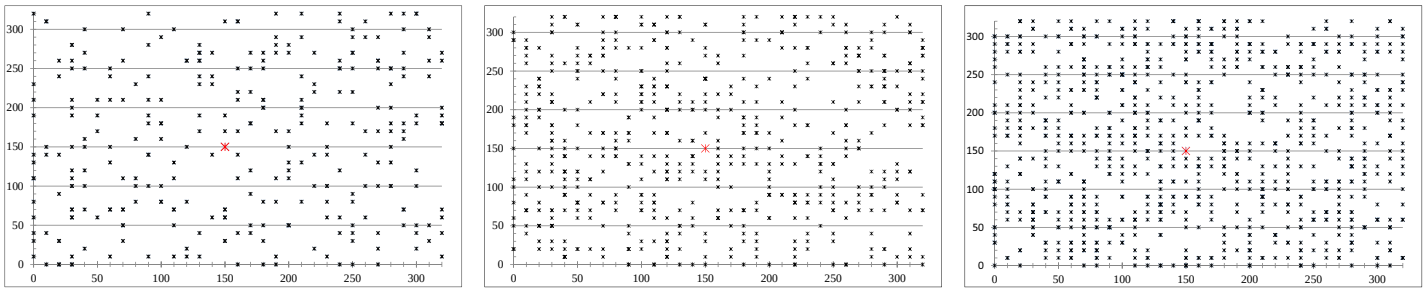
Fig. 4. Avg. msg. size and avg. energy consumption per node (per epoch) for different participation levels (parents/node = at most 3, total nodes = 1024, link error rate = 0.25)

that do not provide fault-tolerance or confidentiality. A 96.2% reduction in the average message size and 94.5% reduction in energy consumption is achieved using the compression optimization. This is due to the fact that the P vector starts from the higher network tracks with more "0" bits than "1" bits and then as the P vectors gets aggregated towards the sink more "1" bits are introduced. Hence, the P vectors are very compressible at both higher and lower level tracks.

2) Effect of Participation Level: A 100% participation rate is not a common case in collaborative WSNs, as often only a fraction of sensor nodes satisfies a query. Most queries involve a conditional clause, such as a WHERE condition. Only those sensors that satisfy the condition are expected to contribute to the query response. This experiment identifies the effect of the participation level on the overall system overhead. In particular, we fix the link error to 0.25, with 1024 sensor nodes deployed in the grid. We report only energy and message size (RMS is relatively insensitive to participation level).

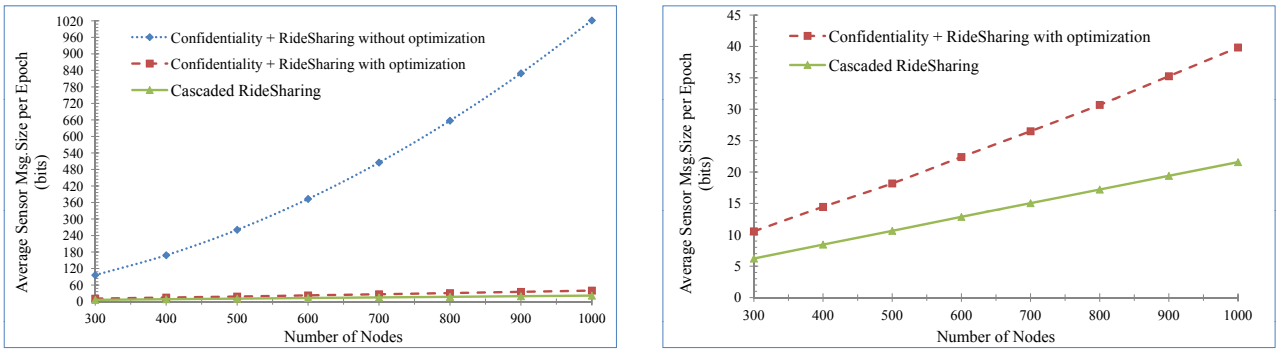
From Figures 4a and 4b, we can see a huge improvement in the overall system overhead when compared with the 100% node participation (cf. Figure 3). Our new scheme with compression optimization shows only an average of 7.1% and 3.6% increase in the average message size and average energy consumption, respectively, when compared with that of Cascaded RideSharing. Note that nodes that do not participate in the query result still need to send the L vectors to propagate the r- and e- bits among backup parents in the same track. This is necessary for the correctness of the fault-tolerance scheme. Note also that with lower participation levels, the P vector contains mostly "0" entries, hence the compression reaches excellent levels. For example, for 25% participation of the sensor nodes, the average message size per node decreases by 45% when compared with 100% participation for the same link error rate.

3) Effect of Network Density: In this experiment, we study the effect of the network density on the system performance for the different schemes. We study this effect by varying the number of sensor nodes from 300 to 1000 within the fixed 320 $\times$ 320 ft^2 grid. Using a uniform random distribution function, each sensor node is assigned a random (x, y) position in the grid. Figure 5 shows sample random deployments of 300, 600 and 1000 nodes in a 320×320 ft^2 grid. Each simulation run then uses a different random network deployment, so the results represent the average over 10 different random



(a) 300 nodes (b) 600 nodes (c) 1000 nodes

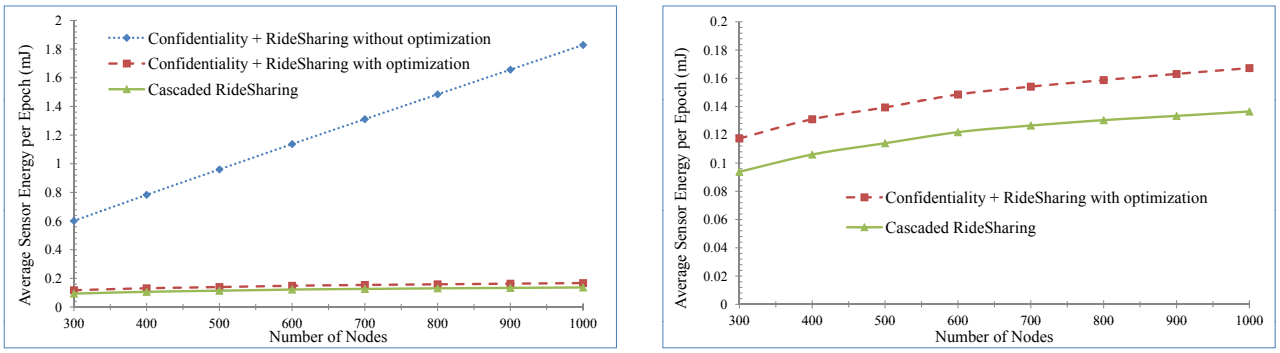
Fig. 5. Example random deployment of nodes in a 320 $\times$ 320 ft^2 grid.



(a) Confidentiality + RideSharing (with/without optimization)/Cascaded RideSharing schemes

(b) Confidentiality + RideSharing (with optimization)/Cascaded RideSharing schemes

Fig. 6. Avg. msg. size per node (per epoch) for different network densities (parents/node = at most 3, participation = 100%, link error rate = 0.25)



(a) Confidentiality + RideSharing (with/without optimization)/Cascaded RideSharing schemes

(b) Confidentiality + RideSharing (with optimization)/Cascaded RideSharing schemes

Fig. 7. Avg. energy consumption per node (per epoch) for different network densities (parents/node = at most 3, participation = 100%, link error rate = 0.25)

deployments. The link error rate is fixed to 0.25, and sensors participation is 100%.

Figure 6a shows the average message size per node as the network density increases, while Figure 6b zooms in on the difference between the compression optimization and the Cascaded RideSharing scheme. Compression optimization reduces overhead: the average message size ranges from 10.5 to 39.8 bits per node (which represents from 60% to 80% overhead over the Cascaded RideSharing scheme), as the network density increases from 300 nodes to 1000 nodes. Figures 7a and 7b shows the average energy consumption overheads as the network density increases. Both figures illustrate the same metric, but Figure 7a shows the improvement of the compression optimization scheme over that without compression. The improvement in terms of average energy consumption per node is on average 90.2%. Figure 7b zooms in on the difference between our scheme with compression optimization and the Cascaded RideSharing scheme. From the figure we can see that the average energy consumption ranges from 0.11 to 0.16 mJ per node (representing an overhead that ranges from 22% to 25%), as the network density changes. It is worth mentioning that very dense deployments – as in the 1000 nodes case – are highly improbable deployments, therefore overheads of up to 25% and 80% of energy and message size,

respectively, represent extremely unlikely situations.

VI. Conclusions

We introduced a confidentiality-preserving and fault-tolerant in-network data aggregation protocol for deployments of collaborative WSNs. Our protocol allows the aggregation of sensor readings while maintaining end-to-end confidentiality of both individual sensor readings and the aggregate result. Our protocol makes use of a simple and efficient additive homomorphic cryptographic scheme and further offers robustness of the aggregation process. The protocol guarantees that with high probability every sensor reading will contribute to the final aggregate through error detection and error correction techniques. Extensive simulations show that our new protocol achieves a high degree of robustness by offering an improvement of 48.2% in the Root Mean Square (RMS) error of the final aggregate result. Energy and messages size overheads are acceptable (about 3% and 7% increases, respectively). For dense network configurations and 100% nodes participation the maximum incurred energy consumption overhead was 25%. In the future, we plan to investigate ways of extending our protocol to also provide end-to-end integrity verification in a confidentiality-preserving manner, which is a more complicated task than the hop-by-hop integrity checking we have discussed earlier. Although schemes exist to provide end-to-end integrity and confidentiality in the presence of malicious nodes (e.g., [12], [16]), these approaches are not tolerant to failures in the underlying aggregation protocol. Achieving this goal will be a significant challenge.

References

[1] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys '03*. New York, NY, USA: ACM, 2003, pp. 1–13.

[2] W. Licheng, W. Lihua, P. Yun, Z. Zonghua, and Y. Yixian, "Discrete logarithm based additively homomorphic encryption and secure data aggregation," *Inf. Sci.*, pp. 3308–3322, August 2011.

[3] F. Armknecht, D. Westhoff, J. Girao, and A. Hessler, "A lifetime-optimized end-to-end encryption scheme for sensor networks allowing in-network processing," *Comput. Commun.*, pp. 734–749, March 2008.

[4] S. Peter, K. Piotrowski, and P. Langendoerfer, "On concealed data aggregation for wireless sensor networks," in *Proceedings of the 24th International conference of the Computer and Communications Societies*. IEEE, 2005.

[5] 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.*, pp. 20:1–20:36, June 2009.

[6] S. Gobriel, S. Khattab, D. Moss, J. Brustoloni, and R. Melhem, "Ridesharing: Fault tolerant aggregation in sensor networks using corrective actions," in *3rd Annual IEEE Communications Society on Sensor, Mesh and Ad Hoc Communications and Networks, SECON'06*, September 2006, pp. 595–604.

[7] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proceedings of the 20th International Conference on Data Engineering*, March 2004, pp. 449–460.

[8] M. Garofalakis, "Proof sketches: Verifiable in-network aggregation," in *IEEE Internation Conference on Data Engineering, ICDE'07*, April 2007, pp. 996–1005.

[9] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *Wireless Communications, IEEE*, vol. 14, no. 2, pp. 70 –87, April 2007.

[10] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: new aggregation techniques for sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04*. New York, NY, USA: ACM, 2004, pp. 239–249.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, pp. 131–146, Dec. 2002.

[12] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proceedings of the 13th ACM conference on Computer and communications security, CCS '06*. New York, NY, USA: ACM, 2006, pp. 278–287.

[13] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Sdap: A secure hop-by-hop data aggregation protocol for sensor networks," *ACM Trans. Inf. Syst. Secur.*, vol. 11, pp. 18:1–18:43, July 2008.

[14] E. Mykletun, J. Girao, and D. Westhoff, "Public key based cryptoschemes for data concealment in wireless sensor networks," in *In IEEE International Conference on Communications, ICC'06*, vol. 5, june 2006, pp. 2288 –2295.

[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptology, EUROCRYPT'99*. Springer-Verlag, 1999, pp. 223–238.

[16] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "Pda: Privacy-preserving data aggregation in wireless sensor networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications*, May 2007, pp. 2045 –2053.

[17] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM '05*. New York, NY, USA: ACM, 2005, pp. 133–144.

[18] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04*. New York, NY, USA: ACM, 2004, pp. 250–262.

[19] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *J. Comput. Syst. Sci.*, vol. 31, pp. 182–209, September 1985.

[20] S. Felsner, G. Liotta, and S. K. Wismath, "Straight-line drawings on restricted integer grids in two and three dimensions," in *Revised Papers from the 9th International Symposium on Graph Drawing, GD '01*. London, UK: Springer-Verlag, 2002, pp. 328–342.

[21] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, pp. 11–25, October 2001.

[22] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis, "Tina: a scheme for temporal coherency-aware in-network aggregation," in *Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access, MobiDe '03*. New York, NY, USA: ACM, 2003, pp. 69–76.

[23] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM conference on Computer and communications security, CCS '03*, New York, NY, USA, 2003, pp. 52–61.

[24] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Trans. Inf. Syst. Secur.*, vol. 8, pp. 228–258, May 2005.

[25] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," in *RFC 2104*. United States: RFC Editor, February 1997.

[26] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04*. New York, NY, USA: ACM, 2004, pp. 188–200.

[27] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGPLAN Not.*, pp. 93–104, November 2000.