Privacy and Robustness for Data Aggregation in Wireless Sensor Networks

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

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications ; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security

Keywords

Fault tolerance, privacy, wireless sensor networks, in-network aggregation

1. INTRODUCTION

Wireless Sensor Networks (WSNs) consist of small sensing and computing devices that have limited power, storage, computation, and network bandwidth capabilities. Queries to the sensor nodes are injected into the network by a data *sink node*. The simplest way to respond to a query is for each sensor node to individually report back its reading to the sink, at which point all of the readings are processed. This unicast delivery requires that intermediate sensors route all such packets to the sink, which can lead to excessive energy consumption due to the large number of individual responses that need to be transmitted.

In-network data aggregation improves the energy efficiency of this process by allowing each node along the routing path to aggregate *all* values received from its children into a *single* response value. This avoids the excessive communication required to route individual sensor readings to the data sink. In-network aggregation paths can typically be classified into one of two categories: *tree-based* or *multipath-based*. The tree-based approach provides the minimal communication overhead by constructing a spanning tree across all sensor nodes to facilitate routing. However, 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, this approach can lead to large errors in the average case. Multipath-based 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 the aggregate value.

In addition to considering the robustness of the aggregation process, many applications (e.g., battlefield surveillance and tracking) require that the confidentiality of individual sensor readings be preserved. Without such a guarantee, adversaries within the proximity of the network could infiltrate the network, eavesdrop, and gain useful information. 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., [6, 7]), or assume perfectly reliable communication links (e.g., [2, 5]).

In this poster, we develop a protocol for reliably carrying out in-network aggregation in sensor networks exhibiting link failures while also maintaining the end-to-end confidentiality of individual sensor readings. Our protocol uses a lightweight homomorphic cryptosystem [3] to enable the 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 [4] to maintain additional metadata that allows the sink node to recover the key needed to decrypt the hidden aggregate value.

2. PRIVACY AND FAULT TOLERANCE PROTOCOL

In this section, we first describe the network and attacker models assumed in this poster, then we briefly discuss the models and building blocks used. Finally we present our new protocol.

2.1 Network and Attack Model

We assume a multi-hop network that consists of n sensor nodes and a single trusted sink node. Each sensor node has a unique identifier ID and shares a unique symmetric key with the sink. These keys are assumed to be pre-shared at deployment time. Sensor nodes are small, battery-operated devices capable of performing simple computations and broadcast communications. The data sink, on the other hand, is a more capable node with higher computational and storage capabilities.

In this poster, we are concerned mostly about the privacy of the sensor readings. We assume a network of two types of attackers: (a) *Honest but curious* sensors that correctly perform the in-network aggregation process, but wish to learn information about the readings of other sensors if at all possible; this is representative of sensor deployments in which individual sensors may become compromised, but continue to function in a seemingly correct manner in order to gather information about the state of the larger network. (b) *Quiet infiltrators* that are able to stealthily infiltrate the network, eavesdrop, and either accumulate the information

CCS'10, October 4–8, 2010, Chicago, Illinois, USA.
ACM 978-1-4503-0244-9/10/10.

gathered or send the information in an undetected way. We assume that an adversary can control any arbitrary number of (colluding) attacker sensor nodes, and can eavesdrop on all communication channels. The sink node is assumed to remain uncompromised.

2.2 Building Blocks

Our privacy preserving scheme makes use of the symmetric key, additively homomorphic stream cipher proposed in [3]. In this cryptosystem, a keyed pseudo-random generator is used to effectively generate per-user keystreams that are used to encipher sensor readings stored as integer values. For example, a sensor node sharing a key k with the sink and using pseudo-random generator g can encrypt its j^{th} reading, 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.

As for our robustness model, we have adopted the Cascaded RideSharing [4] fault tolerance scheme for duplicate sensitive functions. 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. 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 in which each node has access to 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 (towards the sink). Primary parents form a spanning tree that is used in case of error-free operations, while backup parents help compensate for errors in the primary links.

In the RideSharing model, every sensor node transmits its reading to its primary parent according to a predefined TDMA schedule. If the primary parent does not receive any data from a child node in its predetermined time slot, it eventually raises a signal to its neighbors indicating that it did not hear any values from a specific child. In that case, backup parents take the responsibility of aggregating the missing value. Cascaded RideSharing can be thought of as token delegation in which the primary parent initially holds the token and delegates the token in case it does not receive data from a specific child to the next backup parent.

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). As long as no error occurs on the primary edge, the primary parent receives the values of its child, aggregates it, and sets the corresponding r-bit in the L vector for this child to '1'. If an error occurs on the primary edge, the primary parent sets the e-bit to '1' indicating a missing value from this child. Every parent attaches its bit vector to each message it sends. Other (backup) parents within the same track can overhear this L vector and decide whether to take any corrective action by examining the r- and e-bits.

2.3 Protocol Details

At a high level, our approach to providing fault-tolerant and privacy-preserving in-network aggregation works by adding the necessary elements to combine Cascaded RideSharing with the additively-homomorphic stream cipher described in [3]. In the event that the readings of *all* sensor nodes are included in the final aggregate value, combining these algorithms is simple: (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, it is rarely the case that *all* sensor nodes will contribute readings to an aggregate computation. This can occur either because of node- or link-level failures that prevent a sensor's reading from being included in the final aggregate, or simply because not every sensor will have a reading to contribute to every query. In this case, the sink node will 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, our protocol introduces modifications to the Cascaded RideSharing protocol that 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.

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 (PC), a set of child nodes for which this node is a backup parent (BC), the list of peer nodes in this track (SP), and an optional sensor reading to include in the aggregation (m). In addition to maintaining the L vector needed by the Cascaded RideSharing protocol, Algorithm 1 also maintains a *privacy vector*, called the P vector, to keep track of nodes that have successfully contributed to the final aggregate.

The protocol proceeds as follows. If the sensor node has a non-null reading m to contribute to the aggregate, it is first encrypted and then added to the local aggregate A. At this point, 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 to determine the corrective action it needs to take for each child. At this point, 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 received L vector.

After receiving data from all of its child nodes, the sensor transmits its updated aggregate value A, its updated P vector, and its local L vector to its parent nodes (primary and backup) and to the peer (backup) parents.

Algorithm 1: Aggregator

```
input : PC, BC, SP, m
A := 0;
P := 0̄;
L.r := 0̄;
L.e := 0̄;
if m NOT NULL then                    // Aggregate own value
    A := A + m + 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
        else                          // Propagate the error signal
            L[C].e := 1;
        end
    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 XOR 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;
```

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). 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.

3. SIMULATIONS AND EVALUATION

To understand the costs and benefits of our approach we implemented four protocols within the CSIM simulator [1]: (i) a spanning-tree based aggregation protocol that provides neither fault-tolerance nor data confidentiality; (ii) the Cascaded RideSharing protocol, which provides only fault tolerance; (iii) the basic version of our protocol described, which provides both fault-tolerance and data confidentiality protection; and (iv) an enhanced version of our protocol that applies run-length encoding (RLE) to the P vector to minimize data transmission overheads.

All protocols were compared against three main metrics: (a) average relative root mean square error (RMS) of the final aggregate normalized to the correct aggregate result; (b) average energy consumed per node for transmitting, listening, and receiving data; (c) average message size transmitted per node.

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 over the traditional spanning tree schemes for networks with high error rates (up to 35%). The system overheads in terms of average energy consumption and average message size per node are acceptable in representative network settings. Specifically, our simulations show that our protocol incurs only an average of 7.1% and 3.6% increases in the average message size and average power consumption, respectively for different participation levels of the sensor nodes. For dense network configurations and 100% nodes participation the maximum incurred power consumption overhead was 25%.

4. CONCLUSION

In this poster, we presented a privacy-preserving and fault-tolerant in-network data aggregation protocol for wireless sensor networks. Our protocol allows the aggregation of sensor readings while maintaining end-to-end privacy of both individual sensor readings and the aggregate result. This protocol makes use of a simple and efficient additive homomorphic cryptographic scheme and further offers robustness of the aggregation process via modifications to the Cascaded RideSharing fault tolerance scheme. The data sink is the only authorized node capable of retrieving the final plaintext aggregate result. Furthermore, the protocol guarantees that with high probability every sensor reading will contribute to the final aggregate through error detection and error correction techniques. In the future, we plan to investigate ways of extending our protocol to also preserve the integrity of the aggregation process in the presence of faulty or malicious sensor nodes.

5. REFERENCES

[1] "CSIM Simulator", http://www.mesquite.com/.

[2] J. Girao, D. Westhoff, and M. Schneider, "CDA: Concealed data aggregation in wireless sensor networks", in Proc. 40th International Conference on Communiacations, in IEEE ICC, May 2005.

[3] C. Castelluccia, A. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks", ACM Transactions on Sensor Networks, Vol. 5, No. 3, Article 20, May 2009.

[4] S. Gobriel, S. Khattab, D. Mossé , J. Brustoloni, and R. Melhem, "RideSharing: Fault tolerant aggregation in sensor networks using corrective actions", IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks,in SECON, 2006.

[5] W. He, L. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy preserving data aggregation in wireless sensor networks", 26th IEEE International Conference on Computer Communications, May 2007.

[6] Y. Sang, H. Shen, Y. Inoguchi, Y. Tan, and N. Xiong, "Secure data aggregation in wireless sensor networks: A survey", in Proc. of the Seventh International Conference on Parallel and Distributed Computing, in PDCAT, 2006.

[7] E. Mykletun, J. Girao, and D. Westhoff, "Public key based cryptoschemes for data concealment in wireless sensor networks", in Proc. IEEE International Conference on Communications, in IEEE ICC, 2006.