PRIVACY-PRESERVING IN-NETWORK AGGREGATION IN WIRELESS SENSOR NETWORK

Wei Zha and Wee Keong Ng
School of Computer Engineering, Nanyang Technological University, 639798 Singapore, Singapore

Keywords: Privacy-preserving, Wireless Sensor Network, In-network Processing.

Abstract: Ubiquitously deployed wireless sensor networks provide grate conveniences for environment monitoring. However, it also brings the risk of violating privacy. Sensitive sensor data disclosed to malicious part may cause unexpected lost. In this paper, we propose a privacy-preserving in-network aggregation protocol for wireless sensor network based on the concept of data slicing, mixing and merging with a novel share key management scheme. Our protocol allows performing in-network aggregation in sensor network while keeps the privacy of participates. Although we only study additive aggregation in this paper, our protocol can be easily extended to other aggregation functions, including average, count and many other functions based on aggregation as long as these aggregation functions can be reduced to additive aggregation function. Performance evaluation yields the efficiency and effectiveness of our protocol.

1 INTRODUCTION

Wireless sensor networks are gaining more momentum due to its widely military and civilian usages. Major sensor network applications include object tracking, environment surveillance and animal habitats monitoring. Wireless sensor networks may thoroughly change the way people interact with their environments.

Sensor nodes are usually resource constrained to archive low cost. Therefore, it is necessary to maximally utilize the existing resource of each sensor node, such as performing in-network data aggregation to reduce communication and energy cost. In practical, many sensor network applications concern aggregated data readings of a certain region rather than individual sensor reading. Thus, the ability of performing in-network aggregation has received substantial attention. Unless otherwise stated, we use the term "aggregation" stands for both general "aggregation" and "in-network additive aggregation" for brevity.

Usually, a data aggregation function is defined as $f = \sum_{i=1}^{N} d_i(t)$, where $d_i(t)$ is the sensor reading of node i at time t. A simple aggregation in sensor network could be that each sensor node sends back their readings to sink node upon receiving query; and sink node calculates the sum off-line. It is easy but not efficient. To address this issue, aggregation methods have been proposed (Madden et al., 2002; Deshpande et al., 2003; Solis and Obraczka, 2004; Tang and Xu, 2006).

On the other hand, as sensor network applications grow, privacy becomes one of the most challenging issues. Many sensor network applications include sensitive measurements of people daily life. People may not allow that sensor network to be deployed if they find their private information may be disclosed to others. In the following, we brief a motivating application that require privacy-preserving aggregation in wireless sensor networks.

Scenario: Assume National Institute of Health (NIH) wants to measure the average weight of teenagers to analyze the health level of the young generation. Individual weight information can be easily collected by the sensor mounted in their shoes and sent back to NIH. However, selected participants may not wish to disclose their weight information. On the other hand, it is also not necessarily for NIH to know each individual weight information. The sum of participants' weight and the number of participants would be enough. A solution which can satisfy the needs of both NIH and participants are desirable.

From aforementioned scenario, we see the importance of preserving individual privacy. A feasible solution to obtain accurate aggregation while preserving the privacy of individual is desirable. An add-on benefit of private-preserving aggregation includes communication bandwidth and energy conserving as well as enhanced security.

1.1 Related Work

End to end encryption is a well known method in computer network to guarantee private communication as long as the two participates have agreement on public/private keys. However, it is not a good candidate because in-network aggregation cannot be applied. There are some works about privacy-preserving aggregation in sensor network have already be done (Girao et al., 2005; Castelluccia et al., 2005; He et al., 2007; Feng et al., 2008; Zhang et al., 2008; Westhoff et al., 2006; Roy et al., 2006; Shi et al., 2010). Due to space limitation, we only briefly review two works, PDA and PriSense, which are most close to our work. Some details of their works are explained in next section.

PDA (He et al., 2007) proposes a scheme called Slice-Mix-aggRegaTe (SMART) for privacy-preserving aggregation using a random key distribution mechanism. Before sensor nodes deployment, PDA generates a large key-pool and randomly draw k keys for each sensor node. Upon deployment, only the sensor nodes hold the same key can establish a link. When a sensor i is going to send its reading data d_i, it slices d_i into $m+1$ pieces, $d_{i,j}$ where $j \in [1, m+1]$. It keeps one data slice, encrypts and sends out the rest of data slices to those nodes which share the same key. Every node with a data to report performs this process. When this process is finished, each node holds may encrypted date slices. It then decrypts them and adds them together. Now, each node holds a mixed data which does not present any real information. These mixed data can be sent to sink node by using any existing aggregation methods (Yao, Y. and Gehrke, 2002; Krishnamachari, L. and Estrin, D. and Wicker, 2002; Fasolo, E. and Rossi, M. and Widmer, J. and Zorzi, 2007; Madden et al., 2002).

PriSense (Shi et al., 2010) is based on the same slicing, mixing and merging idea but different in share key generating and management. In their scenario, sensor nodes are with mobile ability. Only sensor nodes registered with base-station can report their readings. When a node registers at a base-station, that base-station assigns an ID to that node and it periodically broadcasts the active node IDs of this network. When node A with ID_A wishes to send a data slice to node B with ID_B, it calculates a share key $key_{A,B} = F(H(ID_A), H(ID_B))$, where $H(*)$ is predefined good hash function and $F(*)$ is predefined mapping function. Before sending the real data slice, node A sends a test message to node B which is a random number r encrypted by their share key denoted as $\langle r \rangle_{key_{A,B}}$. If node B can successfully return $\langle r+1 \rangle_{key_{A,B}}$, A considers this link is secure and send a data slice to node B. The rest of steps are exactly the same as in PDA.

1.2 Our Contribution

In this paper, we propose a privacy-preserving in-network aggregation protocol which shares the similar idea as PDA and PriSense on data slicing, mixing and merging concept, but differs significantly in the key generating and management. PDA randomly draws k share keys into sensor node from a large key-pool. Due to memory constrain, value of k cannot be too large. While small k value increases the possibility of network disconnection. PriSense generates share keys on demands. However, malicious parties can easily decrypt encrypted data slices since active node IDs are known to all.

In our protocol, we hire a well known algorithm, Diffie Hellman Key Exchange scheme (Hellman, 2002), to generate share keys on demands. Thus, it is impossible for malicious part to decrypts encrypted data slices without the knowledge of share key. Because two nodes always can make agreement on share key, network disconnection will never happen in our protocol. Our work is more secure than PriSense and more reliable than PDA.

The rest of this paper is organized as follows. In Section 2, we recall the concept of data slicing, mixing and merging and introduce our protocol. Then, we theoretically analyze our protocol and compare it with PDA and PriSense in terms of efficiency and security in section 3. In section 4, experimental results are given. In section 5, we summarize this work and point out some future work.

2 SYSTEM MODEL

Our protocol is based the concept of data slicing, mixing and merging but with a more secure and efficient key management strategy. In this section, we first review this concept, then introduce a different key management scheme.

2.1 Data Slicing, Mixing and Merging Approach

In wireless sensor networks, a data aggregation function is defined as: $f = \sum_{i=1}^{N} d_i$, where d_i is individual sensor reading and N is the number of sensor nodes.

Generally, there are 3 steps in data slicing, mixing and merging approach:

Step 1: Assume node i is going to send a data d_i, it first slices d_i into $m+1$ pieces, $d_i = \sum_{j=1}^{m+1} d_{i,j}$, where $m+1 \leq J$. J is the number of neighbors of node i.

Step 2: Node i keeps one data slice and sends the rest m pieces to its neighbors. Note that, node i sends data slice to its neighbor only if they share the same key. In PDA, each node randomly select m nodes as its receiving nodes from its one-hop neighbors. PriSense has studied different receiving nodes selection strategies, e.g., one-hop, h-hops and random. Their results show that selecting one-hop neighbors as receiving nodes is the most simple and efficient one. In this work, we also randomly choose receiving nodes from one-hop neighbors.

Step 3: Each node waits for a long enough period to make sure there is no more data slice to receive. Then, each node sums all the data slices it has received together with its own data slice (if any), denoted as d'_i, and sends it to sink node. In this step, data is usually sent by a tree structure to perform aggregation. Many existing algorithms can be apply. In this paper, we omit the detail of this step for brevity.

This approach is based on the fact that:

$$f = \sum_{i=1}^{N} d_i = \sum_{i=1}^{N} \sum_{j=1}^{m+1} d_{i,j} = \sum_{i=1}^{N} d'_i$$

An example is shown in Figure 1. Node 1, 3 and 5 have data satisfies a coming query, denoted by gray nodes. Each sensor reading is sliced into 3 pieces. One piece is kept by itself, the rest two pieces are sent to randomly selected one-hop neighbors as illustrated by Figure 1(a). Each node waits for a while to make sure there is no more data slice to receive. Then, each node decrypts received data and mix them with its own data (if any) as illustrated by Figure 1(b). Although the mixed data does not represent individual sensor reading any more, mixed data are still encrypted before sending out for secure purpose. On the way to sink node, medium node can perform aggregation to reduce communication. Take node 5 for instance, instead of sending d'_3, d'_4 and d'_5 to node 6 separately, node 5 sends the sum $d'_3 + d'_4 + d'_5$ to node 6. One possible data merging and aggregation way is shown in Figure 1(c). The final sum at sink node is exactly the sum of $d_1 + d_3 + d_5$ because:

$$d_1 + d_3 + d_5 = \sum_{i \in [1,3,5]} \sum_{j=1}^{3} d_{i,j} = \sum_{i=1}^{6} d'_i$$

2.2 Share Key Set Up and Management

In our protocol, there is a link between node A and B only if they have the same share key $Key_{A,B}$. Data transmitted through this link is encrypted by $Key_{A,B}$ and decrypted by the same key upon receiving. In another words, node A sends out $\langle d_{i,j} \rangle_{Key_{A,B}}$. Upon receiving, node B decrypts $\langle d_{i,j} \rangle_{Key_{AB}}$ with Key_{AB}. The motivation of using symmetric encryption is based on the report of Matt (Eschenauer and Gligor, 2002). For a sensor node such as Motorola MC68328, a 1024*bit* RSA encryption operation costs 42*mJ* energy. While a 1024*bit* AES encryption costs only 0.104*mJ* energy. PDA and PriSense both adopted symmetric encryption in their works.

Different from PDA, our protocol only generates share keys when a node is ready to distribute its data slices. Well known Diffie Hellman Key Exchange scheme (Hellman, 2002) is adopted by our work. There are 3 steps in Diffie Hellman Key Exchange for generating share key between two nodes. We assume node A is going to send node B a data slice $d_{i,j}$:

Step 1: Node A generates a prime P and a generator G, send them to node B in clear text.

Step 2: Node A generates a random number R_A and send the result of $S_A = Mod(G^{R_A}, P)$ to node B. Node B does this simultaneously and sends $S_B = Mod(G^{R_B}, P)$ to node A.

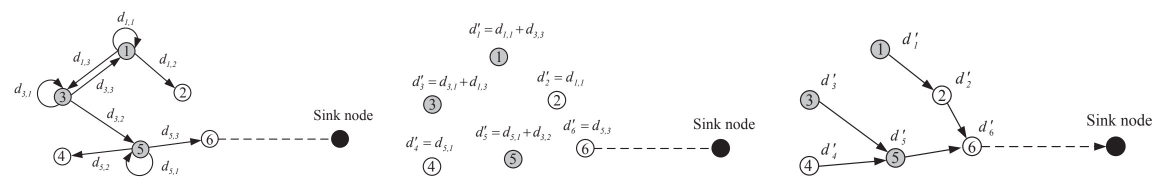
Step 3: Upon receiving, node A can calculate the share key by $Key_{AB} = mod((G^{R_B})^{R_A}, P)$. Node B could obtain the same key by $Key_{AB} = mod((G^{R_A})^{R_B}, P)$.

Thus, share key is known by both node A and B. Node A encrypts data slice $d_{i,j}$ and sends to B. Upon receiving, node B is able to decrypt it correctly.

Note that, it is not necessary for two nodes to make agreement on share key every time. Old share keys can be reused to archive energy conservation. One possible strategy could be every node keeps k share keys with each one-hop neighbor. A node randomly pick up one key from k keys to encrypt data as long as the key is not expired. Whenever the available keys are less than k due to key expiration or other security purpose, it makes a new agreement on share key with the receiving node.

3 COMPARISON AND PERFORMANCE EVALUATION

PDA and PriSense adopt the same data slicing, mixing and merging concept. We compare our work with them in this section and evaluate the performance of our work.



(a) Data slicing and distribution, $m = 2$. (b) Data mixture. (c) Data merging and aggregation.

Figure 1: Data slicing, mixing and merging approach. Nodes with data to send is represented by gray nodes. For simplicity, each node breaks its data into 3 pieces in our illustration.

3.1 Comparison with PDA

As mentioned before, key management adopted by PDA pre-generates a large Key-pool off-line and randomly assigns k keys to each node before deployment. That means the k keys will serve that node for lifetime (Eschenauer and Gligor, 2002). One disadvantage of this key management strategy is unnecessary memory occupancy. The probability of two nodes with the same key is not high. Two nodes usually have a very few overlapping keys. Assume each node share α keys with its neighbors on average, $(k-\alpha)/k\%$ key storage memory is wasted. For the parameters used in PDA, $(k-\alpha)/k > 90\%$. Besides, there is possibility that network is disconnected in PDA.

Another disadvantage of PDA is that once detected the share keys of certain node are disclosed, that node must be revoked, because that node is unable to obtain new share keys. In our work, share keys are generated on demand. Possibly only a few keys will be stored in memory comparing with k keys in PDA. Besides, whenever a share key disclosure is detected, new share key can be generated distributively in real time. Considering the risk of disclosing share key, our key management strategy is more secure.

3.2 Comparison with PriSense

PriSense targets mobile sensor network. Sensor node may join and leave a network due to mobility. When a node joins network, it must register with base-station to obtain an unique active ID. That ID may be resigned to other node if this node leaves network. Base-station periodically broadcasts all the active IDs in the network so that each node can get an overview of this network. When node A with ID_A wants to send a data slice to node B with ID_B, it calculates a share key $key_{AB} = F(H(ID_A), H(ID_B))$, where $H(*)$ is a predefined good hash function and $F(*)$ is a predefined mapping function (Zhang and Fang, 2006). Before sending the actual data, node A generates a random number r, encrypts and sends to node B. If node B can correctly return $\langle r+1\rangle_{Key_{AB}}$ to node A, A sends $\langle d_{i,j}\rangle_{Key_{AB}}$ to B.

Key management strategy in PriSense is much less secure than PDA and our protocol. A malicious node can calculate a share key to decrypt overheard data as long as it knows the IDs of sender and receiver. Since base-station periodically broadcasting active node IDs, it is easy for malicious node to calculate a share key with the knowledge of predefined hash and mapping function.

Additionally, PriSense has studied the receiving node selection strategies, namely, one-hop, h-hop and random selection. According to their experimental results, one-hop receiving node strategy is the most cost efficient one. Both PDA and our work choose the one-hop receiving node strategy by default.

3.3 Performance Evaluation

We evaluate our protocol in terms of disclosure probability and communication cost.

- P_d **(Disclosure Probability):** is the probability that adversaries obtain a node's data.
- E_c **(Communication Cost):** is the total energy consumption for answering one query.

Disclosure Probability Analysis. Suppose a sensor network is consist of N sensor nodes, among which there are n malicious nodes collude to get data d_i from node I. The probability that a node is malicious is $P(M) = n/N$. A node with data d_i slices data into $m+1$ pieces before sending. To get d_i, malicious node must get all the $m+1$ pieces.

To get the m data slices, at least m receivers must be malicious nodes. The probability is $P(M)^m$. The remaining one piece kept by node I can only be calculated when all the data received by I is from malicious node and the receiver of d'_i is also malicious. Assume node I receives J pieces of data before sending out mixed data d'_i, the probability of obtaining that one piece data is $P(M)^{J+1}$ (we assume the worst case that m receiver nodes are different with J sender nodes). In conclusion, at least $m+J+1$ nodes need to collude to get d_i.

$$P_d = P(M)^m \times P(M)^{J+1} = (\frac{n}{N})^{m+J+1}$$

Communication Cost Analysis. In the following of this paper we use Φ to denote the set of nodes with data to send and $|\Phi|$ is the number of nodes. We generally classify communication cost E_c into 3 parts, E_{c1}, E_{c2} and E_{c3}.

E_{c1} represents the communication cost of distributing data slice from nodes in Φ to their one-hop neighbors. Recall the process of Diffie Hellman Key Exchange scheme, node A sends a Prime P and a generator G to node B. Then, A and B exchange some information based on P and G. There are 3 messages occurred in total. We assume these 3 messages are in the same length, denote as l_{key}, for simplicity. With the agreement on share key, A sends B encrypted data slice. We use l_{data} to denote the length of that message. Each node in Φ sends out m pieces of encrypted data slice, we have

$$E_{c1} = |\Phi| \times m \times (3 \times l_{key} + l_{data})$$

E_{c2} represents the communication cost of submitting mixed data. In this process, all the nodes in Φ will submit its data. Due to data slice distribution, some nodes which do not belong to Φ but received data slices also need to submit their data. The probability that a node receiving a data slice is $m/N-1$, and the probability that a node receives at least one data slice is $1-(1-\frac{m}{N-1})^{|\Phi|}$. Among the N nodes, $N-|\Phi|$ nodes are not generating any data but may be chosen as receiving nodes. We denote this kind of nodes by set N_s,

$$|N_s| = (N-|\Phi|) \times (1-(1-\frac{m}{N-1})^{|\Phi|})$$

Nodes in Φ definitely have data to submit. For simplicity, we assume the data to submit is also in the length l_{data}, we have

$$E_{c2} = ((N-|\Phi|) \times (1-(1-\frac{m}{N-1})^{|\Phi|}) + |\Phi|) \times l_{data}$$

In the process of submitting mixed data, some medium nodes will be involved. They neither belong to Φ or N_s. The number of this kind of nodes are totally depends on the aggregation methods. It varies when different methods apply. Since we do not specific any particular aggregation methods in our protocol, we leave it as E_{c3} for simplicity. Thus, we can compute the total communication cost $E_c = E_{c1} + E_{c2} + E_{c3}$.

$$\begin{aligned} E_c = E_{c3} + |\Phi| \times m \times (3 \times l_{key} + l_{data}) \\ + ((N-|\Phi|) \times (1-(1-\frac{m}{N-1})^{|\Phi|}) + |\Phi|) \times l_{data} \end{aligned}$$

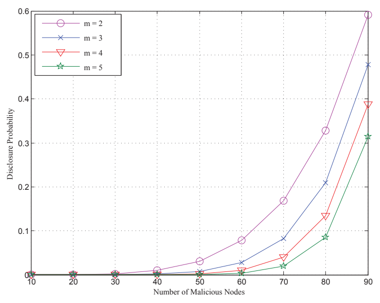


Figure 2: Impact of malicious nodes and m on disclosure probability.

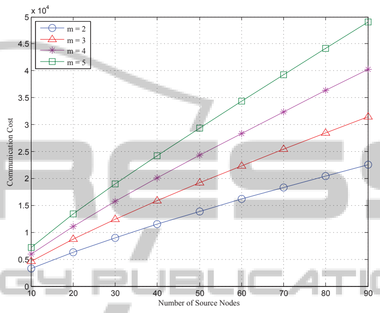


Figure 3: Impact of Φ and m on Communication cost.

4 EXPERIMENTAL RESULTS

In this section, we simulate a wireless sensor network with $N = 100$ nodes. Considering the value exchanged at the second step of negotiating share key could be very huge, we assign $16Byte$ to l_{key} and $50Byte$ to l_{data}. For simplicity, we adopt the same data aggregation method proposed in PDA for step 3 of data slicing, mixing and merging approach. A simple brief of the aggregation method is given here. Upon deployment, each node has a probability p to choose itself as cluster node and send out *HELLO* message. If a node does not decide to be a cluster node, it stores all the *HELLO* messages it has received. When the cluster node selection period is over, each non-cluster node randomly picks up one cluster node based on the *HELLO* messages it has received. Thus, each node sends mixed data to its cluster node. Each cluster node can perform data aggregation before forwarding data to sink node.

We first test the disclosure probability with different number of malicious nodes, then repeat this experiment with different m values. Figure 2 shows the results. We can see that the more malicious nodes are, the higher the disclosure probability is, and vice versa. It is quite obvious that the more malicious nodes, the higher chance a data would be disclosed. We also note that, while the numbers of data slices increase, the disclosure probability decreases. It is easy to understand that the more data slices a node made, the lower chance for malicious nodes to collect all of them. For all the m values, disclosure probability are very low when the malicious nodes are less than 50%. That indicates our method is quite secure even if half of the network is of malicious nodes.

We further record the communication costs with different source nodes Φ and m. The results are illustrated by Figure 3. Communication costs increase while $|\Phi|$ increase. It is intuitive that the more nodes have data to send, the more communication cost will be occurred in the network. It is independent of the number of malicious nodes. Number of data slices m does have a significant impact on communication cost, especially when the number of source nodes Φ is high.

5 CONCLUSIONS

In this paper, we point out the needs of performing privacy-preserving in-network aggregation in wireless sensor networks. Two motivating applications are given. Some existing works have been done. However, they are either not secure enough or overheads of key management is too high. In our work, we reuse their framework but with a more secure and efficient key management methods. The security and communication cost have been thoroughly analyzed. Comparisons with the two most similar works have been made. Experimental results confirmed the correctness of our work. In the future work, we are going to extend our work to other aggregation methods, such as Min/Max. We are also going to study other mutual authentication algorithms with lower computation and communication overhead.

REFERENCES

Castelluccia, C., Mykletun, E., and Tsudik, G. (2005). Efficient Aggregation of Encrypted Data in Wireless Sensor Networks. In *MobiQuitous*.

Deshpande, A., Nath, S., Gibbons, P. B., and Seshan, S. (2003). Cache-and-Query for Wide Area Sensor Databases. In *International Conference on Management of Data*, pages 503–514.

Eschenauer, L. and Gligor, V. D. (2002). A Key-Management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41–47. ACM.

Fasolo, E. and Rossi, M. and Widmer, J. and Zorzi, M. (2007). In-network aggregation techniques for wireless sensor networks: a survey. In *Wireless Communications*, pages 70—-87.

Feng, T., Wang, C., Zhang, W., and Ruan, L. (2008). Confidentiality Protection for Distributed Sensor Data Aggregation. In *International Conference on Computer Communications*, pages 56–60.

Girao, J., Westhoff, D., and Schneider, M. (2005). CDA: Concealed Data Aggregation for Reverse Multicast Traffic in Wireless Sensor Networks. In *IEEE International Conference on Communications*, pages 3044–3049.

He, W., Liu, X., Nguyen, H., Nahrstedt, K., and Abdelzaher, T. T. (2007). PDA: Privacy-Preserving Data Aggregation in Wireless Sensor Networks. In *IEEE International Conference on Computer Communications*, pages 2045–2053.

Hellman, M. E. (2002). An Overview of Public Key Cryptography. In *IEEE Communications Magazine*, pages 42–49.

Krishnamachari, L. and Estrin, D. and Wicker, S. (2002). The impact of data aggregation in wireless sensor networks. In *International Conference on Distributed Computing Systems Workshops*, pages 575—-578.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2002). TAG: a Tiny AGgregation service for adhoc sensor networks. In *ACM SIGOPS Operating Systems Review*, pages 131–146.

Roy, S., Setia, S., and Jajodia, S. (2006). Attack-Resilient Hierarchical Data Aggregation in Sensor Networks. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 71–82. ACM.

Shi, J., Zhang, R., Liu, Y., and Zhang, Y. (2010). Prisense: privacy-preserving data aggregation in people-centric urban sensing systems. In *INFOCOM*, pages 1–9. IEEE.

Solis, I. and Obraczka, K. (2004). The Impact of Timing in Data Aggregation for Sensor Networks. In *IEEE International Conference on Communications*, volume 6, pages 3640–3645. IEEE.

Tang, X. and Xu, J. (2006). Extending Network Lifetime for Precision-Constrained Data Aggregation in Wireless Sensor Networks. In *IEEE International Conference on Computer Communications*, volume 6, pages 1–12.

Westhoff, D., Girao, J., and Acharya, M. (2006). Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution, and Routing Adaptation. In *IEEE Transactions on Mobile Computing*, pages 1417–1431. IEEE Computer Society.

Yao, Y. and Gehrke, J. (2002). The cougar approach to in-network query processing in sensor networks. In *SIGMOD*, pages 9—-18.

Zhang, W., Wang, C., and Feng, T. (2008). GP2S: Generic Privacy-Preservation Solutions for Approximate Aggregation of Sensor Data (concise contribution). In *IEEE International Conference on Pervasive Computing and Communications*, pages 179–184. IEEE.

Zhang, Y. and Fang, Y. (2006). ARSA: An Attack-Resilient Security Architecture for Multihop Wireless Mesh Networks. In *IEEE Journal on Selected Areas in Communications*, pages 1916–1928. IEEE.