DyDAP: A Dynamic Data Aggregation Scheme for Privacy Aware Wireless Sensor Networks

Sabrina Sicari

Dipartimento di Informatica e Comunicazione, Università degli Studi dell'Insubria, via Mazzini, 5 - 21100 Varese, Italy

Luigi Alfredo Grieco

Politecnico di Bari, via Orabona, 4 - 70125 Bari, Italy

Gennaro Boggia

Politecnico di Bari, via Orabona, 4 - 70125 Bari, Italy

Alberto Coen-Porisini

Dipartimento di Informatica e Comunicazione, Università degli Studi dell'Insubria, via Mazzini, 5 - 21100 Varese, Italy

Abstract

End-to-end data aggregation, without degrading sensing accuracy, is a very relevant issue in Wireless Sensor Networks (WSN) that can prevent network congestion to occur. Moreover, privacy management requires that anonymity and data integrity are preserved in such networs. Unfortunately, no integrated solutions have been proposed so far, able to tackle both issues in a unified and general environment. To bridge this gap, in this paper we present an approach for dynamic secure end-to-end data aggregation with privacy function, named DyDAP. It has been designed starting from a UML model that encompasses the most important building blocks of a privacy-aware WSN, including aggregation policies. Furthermore, it introduces an original aggregation algorithm that, using a discrete-time control loop, is able to dynamically handle in-network data fusion to reduce the communication load. The performance of the proposed scheme has been verified using computer simulations, showing that DyDAP avoids network congestion and therefore improves WSN estimation accuracy while, at the same time, guaranteeing anonymity and data integrity.

Keywords: Wireless sensor networks, privacy, anonymity, end-to-end data aggregation, congestion control

Email addresses: `sabrina.sicari@uninsubria.it` (Sabrina Sicari), `a.grieco@poliba.it` (Luigi Alfredo Grieco), `g.boggia@poliba.it` (Gennaro Boggia), `alberto.coenporisini@uninsubria.it` (Alberto Coen-Porisini)

1. Introduction

Wireless Sensor Networks (WSN) technologies support data collection and distributed data processing by means of very small sensing devices [1] with limited computation and energy capabilities. WSN are used in many contexts such as telemedicine, surveillance systems, assistance to disabled and elderly people, environmental monitoring, localization of services and users, industrial process control, and systems supporting traffic monitoring/control in urban/suburban areas, military and/or anti-terrorism operations.

Data transmission is one of the most power demanding tasks in WSN and therefore messages forwarding policies and processing techniques should be carefully designed to extend network lifetime. For example, a very common technique consists in keeping WSN nodes silent as long as no relevant information is detected in the monitored area [2]. Then, when some information is detected, wireless interfaces are turned on and sensors transmit the sensed data. However, the main drawback of such an approach is that it can cause network congestion [3] and as a consequence some information may get lost.

To avoid network congestion one solution consists in using proper in-network algorithms (e.g., see [4, 5, 6, 7]) that can reduce significantly the number of bytes exchanged across the WSN by aggregating data [8, 9, 10]. In fact, in many situations, what is needed are aggregated measures, such as the average temperature of a region, the average humidity, and so on.

Another important issue in WSN is represented by privacy that may be violated by tampering of sensors and/or traffic due to the nature of the wireless channel and its deployment in uncontrolled environments. Thus, privacy aware mechanisms are crucial for several WSN applications such as localization and telemedicine. Among the different aspects characterizing privacy, anonymity is an important requirement for a privacy aware system that aims at protecting the identity of the individuals whose data are handled by the system. Moreover, it may be necessary to take into account privacy also in some application contexts in which data referring to individuals are not directly handled by WSN. For example, in home networks, sensor nodes may collect a large amount of data that may reveal habits of individuals, violating in this way their privacy.

However, the low power resources and the limited computational and storage capabilities of sensor nodes impose severe constraints on how privacy requirements can be satisfied.

Considering together data aggregation and privacy issues is a very challenging problem, falling in the main research area of secure WSN [11]. Many solutions addressing at the same time aggregation and security aspects such as confidentiality, integrity, authentication, and availability can be found in literature (for an exhaustive and very comprehensive view of this topic see [12]). However, to the best of our knowledge, no solution is able to encompass privacy and end-to-end secure data aggregation.

This paper presents an approach that couples a privacy management policy with an original aggregation algorithm able to deal with end-to-end encrypted data. The approach, named DyDAP (*Dy*namic *D*ata *A*ggregation with *P*rivacy functions), is based on the privacy model proposed in previous works [13, 14, 15]. Such a model, defined in UML [16, 17], represents a general schema that can be easily adopted in different contexts. DyDAP adapts such a schema to the context of WSN by introducing concepts, such as nodes, data, actions that are needed to define a privacy policy along with the existing relationships among them. Moreover, DyDAP integrates an original aggregation algorithm designed by exploiting linear discrete time control theory [10, 18]. Using DyDAP, each node periodically evaluates the amount of data to aggregate in order to control the level of its transmission queue. The aggregation process, which merges spatial correlated data working on encrypted information, involves only linear operations and allows the sink node to estimate the confidence level of the aggregated data.

The main goals fulfilled by our approach are: (i) anonymity management; (ii) data integrity check; (iii) data aggregation to reduce the network load; (iv) end-to-end secure data aggregation. The effectiveness of the approach has been studied using computer simulations.

Notice that simulations are used to evaluate aspects such as congestion, data loss, number of transmitted messages, accuracy, while they are not useful to show that security/privacy goals are satisfied. In fact for these latter aspects we rely on the fact that the techniques used (such as encryption) guarantee "by construction" that such goals are met. In conclusion we show that DyDAP avoids network congestion and improves WSN estimation accuracy, while, at the same time, guaranteeing anonymity management and data integrity.

The rest of the paper is organized as follows. Section 2 introduces the foundations for modeling privacy in the context of WSN. Section 3 describes the reference scenario and the foundations on which DyDAP is based. Section 4 presents DyDAP in detail, while Section 5 reports the simulation results that show the effectiveness of DyDAP. Section 6 presents the most relevant related works. Finally, Section 7 draws some conclusions and provides hints for future works.

2. Modeling privacy policies for wireless sensor networks

A privacy policy defines the way in which data referring to individuals can be collected, processed, and diffused according to the rights that individuals are entitled to.

The rest of the paper adopts the terminology introduced by the EU directive [19]. Notice that the terminology is as much technology-independent as possible and therefore, beside the original definition, we provide the needed refinements in order to support the definition of privacy mechanisms in WSN communications:

- *Personal data* means any information related to an identified or identifiable natural person (referred to as *data subject* or *subject*). In the context of WSN, they represent the data sensed by the nodes of the network; in other words, nodes play the role of *subjects* since they receive information from the environment in which they are located.

- *Processing of personal data* (*processing*) means any operation or set of operations which is performed upon personal data, whether or not by automatic means, such as collection, recording, organization, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, blocking, erasure or destruction. In the context of WSN, activities such as sensing data, receiving/transmitting messages, aggregating data, encrypting/decrypting data and verifying data integrity can be considered as *processing.*

- *Controller* means the natural or legal person, public authority, agency or any other body which alone or jointly with others determines the purposes and means of the processing of personal data. In a WSN, nodes play the role of controllers of the network, since they verify the processing actions that involve sensed data.

- *Processor* means a natural or legal person, public authority, agency or any other body which processes personal data on behalf of the controller. In a WSN, the role of processor is played by the nodes of the network.

- Data *subject consent* (*consent*) means any freely given specific and informed indication of his/her wishes by which the data subject signifies his/her agreement to personal data relating to him/her being processed.

As a distinctive feature of a privacy policy, the processor is required to state for what *purpose* data are processed. A purpose can be defined either as a high-level activity (e.g., “monitoring”, “tracking”) or as a set of actions (e.g., “determine the average temperature”, “evaluate the humidity”).

In addition processing actions may be executed under specific *obligations.* An obligation is a set of actions that the processor and/or the controller guarantees to carry out at the end of the processing activities. For example, a node that measures the temperature of the ground may be required to send an alert message whenever the temperature is less than a given threshold.

In a privacy aware system, subjects have to grant their consent before any processing can occur on their data. In the context of WSN we assume that the consent is implicitly given by the nodes of the network. This means that every node accepts that its data may undergo different processing activities each of which consisting in a set of processing actions (and obligations) that may occur on the other nodes of the network. Notice that the above assumption requires that the system modeler adopts adequate mechanisms to assure that nodes can trust each other.

2.1. The UML Model

Starting from the general definitions, a conceptual model of privacy has been provided, using UML, in previous works [13, 14, 15]. In the following we provide a short overview of such a conceptual model along with the refinements needed to take into account the specifities of WSN. The structural aspects are defined using UML classes and their relationships such as associations, dependencies, and generalizations. Fig. 1 depicts a class diagram that provides a high level view of the basic structural elements of the model, while Fig. 2 shows a refined class diagram in which all the needed concepts are introduced.

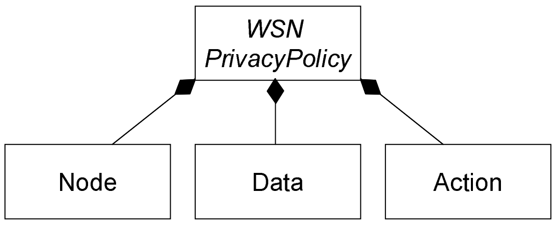


Figure 1: The privacy policy class diagram.

A *WSN_Privacy Policy* consists of three types of classes: *Node*, *Data*, and *Action*. Thus, an instance of class *WSN_PrivacyPolicy* is characterized by specific instances of *Node*, *Data*, and *Action*, and by the relationships among such entities. In what follows we focus on such basic classes.

1) Node represents a member of the network either interested in processing data or involved by such a processing. Nodes are characterized by functions and roles. More specifically:

- *Role* [20] is a key concept of this approach used to characterize nodes with respect to privacy. Thus, we introduced three distinct classes extending class *Role* to represent the three different roles a node can play: *Subject*, to represent nodes that sense data; *Processor*, to represent nodes that process data by performing some kind of action on them (e.g., transmission, forwarding, aggregation, etc.); *Controller*, to represent nodes that verify the actions executed by processors. Notice that a node can play more than one role.

- *Function* represents the task performed by a *Node* in the network in which it operates (e.g., data sensing, message transmission, message forwarding, data aggregation, etc.).

2) Data describes the information referring to subjects that is processed by processors. Class *Data* is extended by means of class *Identifiable*, which represents the information used to uniquely identify nodes (e.g., node identifier), and class *Sensed*, which represents the information that is sensed by the nodes of the network (e.g., temperature, pressure). In turn, class *Sensed* is extended by class *Sensitive* to represent sensed information that deserves particular care

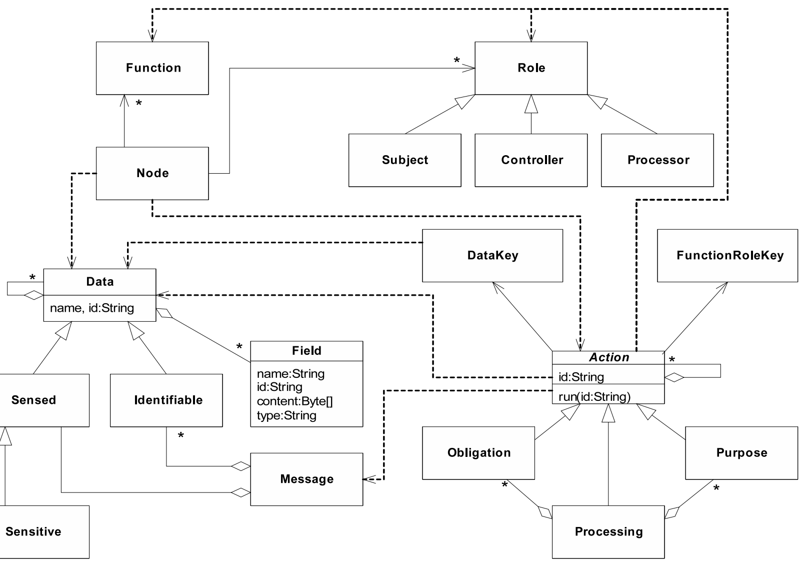


Figure 2: The WSN privacy class diagram.

and that should not be freely accessible. For instance, in telemedicine applications a sensitive datum is the heart beatrate sensed by nodes positioned on the body of patients. Notice that also some common sensed data may deserve particular care. For example, consider a wireless meter reading system used to monitor the environmental parameters of a building comprising several sensor units that communicate information on the current temperature and humidity of the rooms in which they are installed. Although the data sensed by the nodes can not be classified as sensitive, they can be used to gather information on the personal habits of people living in the building. For instance, slight increments of temperature or humidity may indicate the presence of one or more persons in a room. Thus, analyzing such information a burglar can figure out when (part of) the building is empty. Notice that *Data* is a complex structure composed of basic information units, named *Fields*, each of which represents a partial information related to the whole data structure. In turn, instances of *Data* are aggregated into instances of *Message*. Thus, class *Message* represents the basic communication unit exchanged by the nodes of the network and may contain both *Identifiable* and *Sensed* data along with plain data (i.e., instances of class *Data* that do not represent identifiable nor sensed data). The actual structure of messages is discussed in depth in Section 4.

3) Action, represents any operation performed by a node. It is extended by classes *Obligation*, *Processing*, and *Purpose*. Since processing is executed under a purpose and an obligation, *Processing* specifies an aggregation relationship with *Purpose* and *Obligation*. Moreover, each action can be recursively composed of other actions. The most common actions in the context of WSN are defined in what follows:

- *Sensing*: acquisition of data carried out by a node, concerning a specific feature of the system. For instance, the value of the temperature of the room in which the node is installed.
- *Transmission*: sending a message containing data that were sensed (or aggregated) by the current node.
- *Reception*: receiving a message sent by another node of the network.
- *Forwarding*: sending a message previously received by the node, that is the message contains data that were sensed (or aggregated) by other nodes of the network.
- *Verification*: checking the integrity of the data contained in a received message. In this case it may be useful to associate such an action with one or more obligations in order to model the fact that whenever an inconsistency is found some countermeasures, such as sending an alert message to the sink, are taken.
- *Data aggregation*: aggregating received and/or sensed data.

In order to guarantee data confidentiality and integrity as well as to assure that only authorized nodes can access data and/or execute actions, the model defines encryption mechanisms. More specifically, two classes representing encryption keys, named *DataKey* and *FunctionRoleKey*, are introduced. The former class is used to protect the data content of messages, and therefore each node of the network owns a (possibly) different *DataKey* to encrypt the data content of its messages. The latter class is used to guarantee that message communication and data handling are executed only by authorized nodes. In fact, each action is associated with a *function-role* pair and therefore requires a given *FunctionRoleKey* in order to be executed. Thus, only nodes having the corresponding *FunctionRoleKey* can execute a given action. Notice that since a node may play different functions and roles, it may own more than one function-role key.

3. The DyDAP Approach

Let us consider a dense network composed of N nodes, all of which sensing the same type of data (e.g., temperature, humidity). Moreover, nodes can exchange messages among them and therefore sensed data (possibly aggregated) eventually reach the sink [21]. In other words, each node can act as a source of data and/or as a forwarding element. In the former case traffic is locally generated at the sensor node, whereas in the latter case traffic comes from other nodes. The network architecture is shown in Fig. 3.

We assume that nodes are aware of their own geographical position and that they implement a CSMA-like MAC protocol for accessing the wireless channel [22]. The broadcast nature of wireless channel enables a node to determine, by overhearing the channel, whether its messages are received and forwarded by

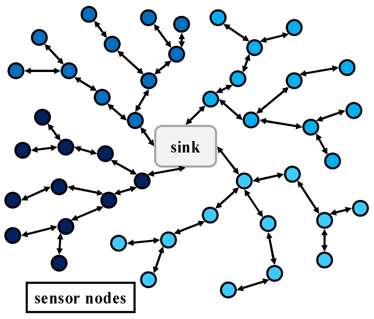


Figure 3: Network topology.

its neighbors [23]. Moreover, we assume that nodes are equipped with a transmission buffer, in which messages may be queued before being transmitted over the network and with a storage table where messages can be stored (the utility of this feature is explained in the next Section). Whenever the node message input rate, which is given by local plus transit traffic rates, exceeds the message forwarding rate at MAC layer (which depends on the MAC protocol, the radio channel and the network topology), messages are queued in the transmission buffer. Thus, if no countermeasure is taken upon the saturation of the message forwarding rate, the transmission queue builds up till its maximum limit and, as a consequence, message losses arise due to traffic congestion. Message losses have two negative effects: (i) they waste energy resources; (ii) they can severely impair the estimation accuracy of the system. Since it is very difficult to regulate the message forwarding rate, the reduction of the input traffic rate is the only possible reaction to a network congestion (as in wired networks [24]).

The approach presented in this paper exploits data aggregation at node level to avoid traffic congestion: when the transmission queue builds up, data therein are aggregated in order to keep the queue length under its maximum limit. Notice that, as far as network congestion is concerned, this is equivalent to reduce the node message input rate.

Aggregating data is not equivalent to collecting individual sensor readings and thus it may not be used in applications in which the latter are necessary (e.g., perimeter control). However,in many applications (such as temperature or sysmic activity monitoring systems) the emphasis is on statistical data and therefore data aggregation can be used to avoid traffic congestion. For example, if the goal is to determine the mean or the variance of sensed data, aggregation can be carried out by simply adding up values[1].

In order to meet privacy and end-to-end security requirements, encryption techniques are used, that is the content of all messages is encrypted before being transmitted over the network. However, the main drawback of this approach is

[1]To compute the mean it is necessary to add up the measured data, while for variance it is necessary to add up also the square of such values [6].

that it may be necessary to decrypt messages before any aggregation process may take place. To overcome such a problem it is possible to use homomorphic stream ciphers such as the one presented by Castelluccia et al. [6], which allows efficient aggregation of encrypted data. In fact, an homomorphic encryption scheme allows arithmetic operations to be performed on ciphertexts. For instance, the addition of two ciphertexts followed by a decryption operation yields the same result as the addition of the two corresponding plaintext values. In this way it would be possible to aggregate encrypted messages without decrypting them in advance.

In what follows we discuss the theoretical foundations on which the DyDAP framework is built, by presenting the way in which the transmission queue is controlled and by shortly reporting how the algorithm of Castelluccia et al. works.

3.1. Controlling the transmission queue

In order to keep the overall network load under a reasonable level, messages in the transmission buffer are aggregated as soon as the queue length starts building up. Messages, generated by the aggregation process, are then forwarded through the network and can be further aggregated by next nodes along the path to the sink. The total amount of data to aggregate is periodically computed by each node using a control theoretic approach [18] that drives the transmission queue toward a constant reference point. The selection of messages to aggregate and the aggregation procedures will be described later on (see *Data Aggregation Protocol*).

Let $\varphi(t_j)$ be the queue length associated with a transmission buffer at the time instant t_j, which represents the beginning of the j-th sampling interval. In the discrete-time domain, its dynamics can be modeled by the following linear model:

$$\varphi(t_{j+1}) = \varphi(t_j) + \xi(t_j) - u(t_j) + u_V(t_j) \tag{1}$$

where, with reference to the j-th sampling interval, $\xi(t_j) \geq 0$ is the number of messages enqueued, $u(t_j) \geq 0$ is the number of transmitted messages; $u_V(t_j) \leq 0$, computed by DyDAP models the control action, i.e., its absolute value $|u_V(t_j)|$ is the number of messages that have to be removed from the queue using aggregation techniques.

More specifically, $\xi(t_j)$ is a function of the input traffic load at the queue, generated from the source nodes, and it may change with time; instead, $u(t_j)$ depends on the MAC protocol. Both $\xi(t_j)$ and $u(t_j)$ affect the queue length in a way that cannot be forecast in advance. For this reason, they will be modeled as disturbances [25].

To avoid buffer overflows, at each sampling time t_j, it is necessary to compute the amount $|u_V(t_j)|$ of messages that should be virtually removed from the queue. The approach is based on the feedback control loop pictured in Fig. 4. In such a scheme, the reference queue level φ_T is compared against the actual queue level $\varphi(t_j)$ and the resulting error $e(t_j) = \varphi_T - \varphi(t_j)$ is fed into a Proportional Integral (PI) regulator, with transfer function $G_c(z)$, to compute

$u_V(t_j)$. A PI controller has been chosen because it is able to filter out the continuous component of the disturbance $\xi(t_j) - u(t_j)$ [18, 25].

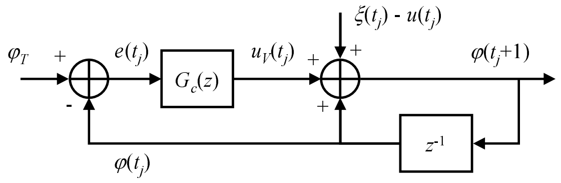


Figure 4: Block diagram of the controlled system.

The transfer function of the PI controller, computed as the $\mathcal{Z}$-transform of its discrete-time pulse response [18], is given by

$$G_C(z) = \beta \cdot \left(1 + \frac{1}{T}\frac{z}{z-1}\right) \tag{2}$$

where β and T are non negative constants.

To tune the PI regulator, the following theorem has been derived.

Theorem 1. *The system, shown in Fig. 4, is asymptotically stable if and only if the following inequalities are satisfied:*

$$\begin{cases} 0 < \beta < 2 \\ \frac{\beta}{T} > 0 \\ 4 - 2\beta - \frac{\beta}{T} > 0 \end{cases} \tag{3}$$

Proof: *The demonstration can be easily derived by applying the Jury criterion [18] to the characteristic polynomial $p(z) = T(z-1)^2 + \beta(T+1)z - \beta T$ of the control system in Fig. 4.*

3.2. Privacy in end-to-end secure aggregated data

The algorithm of Castelluccia et al. [6] is based on a simple and secure additively homomorphic stream cipher that allows efficient aggregation of encrypted data. Homomorphic encryption schemes are especially useful in scenarios where someone, having no decryption keys, needs to perform arithmetic operations on a set of ciphertexts.

The cipher uses modular additions and is therefore very well suited for CPU-constrained devices like sensors. Moreover, aggregation based on this cipher can be used to efficiently compute statistical values such as mean, variance, and standard deviation of sensed data, enabling significant bandwidth gain.

The main idea of [6] is to replace the XOR (Exclusive-OR) operation, typically found in stream ciphers, with modular addition. For reader convenience, we will briefly sketch the additively homomorphic encryption scheme proposed in [6] by applying it in the context of WSN.

Let us consider a network comprising N nodes, each of which is uniquely identified by a label n_i, $1 \leq i \leq N$. Moreover let d_i denote an integer number representing the data generated by node n_i, where $d_i \in [0; M-1]$ and M is a large enough integer, whose value is discussed later on.

Moreover, each node n_i owns a different key $k(n_i, fr)$, where fr represents the Function-Role pair played by n_i. According to the model presented in Section 2 $fr \in \{SS, AP, TP, NC\}$, where SS represents *Sensing-Subject*, AP represents *Authenticator-Processor*, TP represents *Transmitter-Processor* and NC represents *Notifier-Controller*. For instance, the *Sensing-Subject* key of node n_1 is represented by $k(n_1, SS)$. Finally, let Enc and Dec denote the encryption and decryption functions, respectively. Enc (Dec) takes as arguments the plaintext to be encrypted (the ciphertext to be decrypted) and the key.

Thus, the encrypted ciphertext of datum d_i generated by node n_i is given by

$$c_i = Enc(d_i, k(n_i, SS)) = (d_i + k(n_i, SS)) \pmod M. \tag{4}$$

The aggregation of J different ciphertexts $c_1, ..., c_J$ received from other nodes is carried out in the following way:

$$c_{aggr} = \sum_{i=1}^{J} c_i \pmod M \tag{5}$$

Since the above encryption scheme is additively homomorphic, we have that if $c_1 = Enc(d_1, k(n_1, SS))$ and $c_2 = Enc(d_2, k(n_2, SS))$ then $c_1 + c_2 = Enc(d_1 + d_2, k(n_1, SS) + k(n_2, SS))$.

As a consequence, the cleartext of the aggregated data d_{aggr} can then be obtained by:

$$d_{aggr} = Dec(c_{aggr}, k) = c_{aggr} - k \pmod M; \qquad k = \sum_{i=1}^{J} k(n_i, SS)\ . \tag{6}$$

Notice that in order to ensure correctness of the above schema M should be larger than $\sum_{i=1}^{N} d_i$. In fact, if such a constraint is satisfied, decryption will result in a value d^* that is smaller than M. In practice, if $p = \max\{d_i\}$, then M should not be smaller than ($M = 2^{\log_2(p \cdot N)}$).

The keystream k can be generated using a streamcipher, such as RC4, keyed with a node secret key and a unique message. Finally, each sensor node shares a unique secret key with the *sink* of the WSN. Such keys are derived from a master secret (known only to the *sink*) and distributed to the sensor nodes. However, the key distribution protocol is outside the scope of this work.

4. The DyDAP Framework

This section presents the integrated framework DyDAP whose goal is to provide an end-to-end data aggregation scheme with privacy capabilities for WSN. Thus, in what follows we provide an throrough discussion of the message structure, the protocols and the algorithms used in DyDAP to meet the above mentioned goals.

4.1. Message structure

A message is the basic communication unit exchanged by the nodes of the network, that is a message refers to a single transmission hop between adjacent nodes. In what follows messages are denoted by $m_{n,q}$, where n indicates the node that generated and transmitted the message, while q uniquely identifies the message among those generated by node n.

According to the model described in Section 2 messages may contain data the can be further classified as *Identifiable* or *Sensed*. *Identifiable* data include the information that can be used to identify a node, while *Sensed* data include all information sensed by the nodes. Finally, in what follows we refer to data being neither *Identifiable* nor *Sensed* as plain data.

A sensed data, before reaching the sink, passes through different nodes of the network (multi-hop communication) by means of different messages. In order to guarantee the integrity and confidentiality of the end-to-end communication, we use a message structure that keeps track of both the last and the second last hops of the transmission.

A DyDAP message $m_{n,q}$ is defined as a tuple in which all fields, unless otherwise specified, are ciphered.

$$m_{n,q} = \langle curr, prev, sub, data, varD, pos, varP, encP, err, sensId, errId, idList\rangle$$

where:

- *curr* is the couple (n, q) that identifies the current message among those transmitted over the network.
- *prev* is the couple (n_p, q_p), where n_p is the node that operated the second last forwarding of the sensed data contained in the current message, and q_p identifies such a message among those transmitted by n_p.
- *sub* is the couple (n_s, q_s), where n_s identifies the *Subject* node, that is the node that either sensed or aggregated the data, and q_s identifies such a message among those transmitted by n_s. Notice that in case of error notification (see *Reception and Integrity Verification Protocol*) n_s identifies the node that discovered the error.
- *data* is the data either sensed or aggregated by the node described by *sub*.
- *varD* is the variance associated with *data*.
- *pos* is the couple (x, y) describing the geographical position of the node described by *sub*. This field is in clear.
- *varP* is the variance associated with *pos*. This field is in clear.
- *encP* is a ciphered field containing the same information as *pos*. It is included in order to detect any modification of *pos* possibly made by a malicious node.

- *err* is set to 1 whenever an error is detected, otherwise it is set to 0.
- *sensId* is the couple (n_s, q_s) that in case of error notification identifies the node that either sensed or aggregated the correct data and the identifier of the message transmitted by such a node.
- *errId* is the couple (n_m, q_m) that identifies the node that generated the error message and the identifier of the message reporting the error transmitted by such a node.
- *idList* is a list of nodes that processed the original data. It is used by the sink in order to identify the keys required in the decryption process of *data*. Depending on the Maximum Transfer Unit (MTU) of the underlying communication technology, the size of this list is upper bounded. For now on, we will refer to id^M as the maximum number of IDs that can be concatenated to form the *idList*.

Notice that *sensId* and *errId* are used only in case of error notification, i.e., when *err* is set to 1, and that the encryption of the fields *curr*, *prev* and *sub* besides supporting integrity, guarantees anonymity.

Referring to the model of Section 2, fields *curr*, *prev*, *sub*, *pos*, *varP*, *encP*, *sensId*, *errId* and *idList* are instances of class *Identifiable*, while *data* and *varD* of class *Sensed*. The remaining fields are instances of class *Data*, that is they represent plain data. Moreover, *curr*, *prev*, *sub*, *pos*, *encP*, *sensId*, *errId* are composed of two different instances of class *Field*, while *data*, *varD*, *varP* and *err* contain one instance of class *Field*.

In what follows we introduce two possibly different encrypting (decrypting) functions denoted by *Enc* and Enc^* (*Dec* and Dec^*). Functions *Enc* and *Dec* must provide an homomorphic encryption schema as introduced in Section 3, while functions Enc^* and Dec^* may not[2].

4.2. DyDAP protocols

Messages are transmitted over the network by nodes sensing data from the environment in which they are located. Messages are then received by other nodes (at 1 hop distance) that may forward them, verify their integrity or aggregate a set of received messages, generating in this way a new message.

Nodes store part of the messages in their local table before transmitting them over the network. In particular the fields stored in the local table, using field *curr* as hash key, are *prev*, *sub*, *data* and *encP*. This is done to allow a node, upon reception of a message, to identify whether the received message was previously transmitted by the node itself as discussed later on (see Sec. 4.2.2).

Notice that *stored messages*[3] are removed from the local table after a fixed time that has to be greater than $S * \Delta_T$, where S denotes the size of the table

[2]This means that *Enc* and Enc^* (*Dec* and Dec^*) may or may not be the same function
[3]A stored message is the part of a transmitted message that is locally stored

and Δ_T the maximum time elapsing between the generation of two consecutive messages. Since the size of stored messages size is much smaller than the size of transmitted messages, it is very unlikely that the local table overflows.

In this section we present the following protocols:

- *Sensing*, which defines the actions that a node carries out when sensing data.
- *Message Reception and Integrity Verification*, which defines the actions carried out when receiving a message.
- *Data Aggregation*, which comprises the actions that a node carries out to aggregate received messages into a new message.

4.2.1. Sensing

Let n be a node sensing a data d from the environment where it is located. Moreover, let us assume that the position of n is represented by the coordinates (x_n, y_n). According to the function-role classification, when sensing d the node acts as a *Sensing-Subject* (SS) and therefore the node encrypts d using the key $k(n, SS)$. Let q denote the number of messages that n already transmitted over the network. Thus, the message $m_{n,q+1}$ is prepared according to the structure discussed in the previous section. Notice that, when preparing the message the node acts as a *Transmitter-Processor* (TP) and therefore all the cyphered fields but *data* are encrypted using the key $k(n, TP)$.

$$m_{n,q+1} = \langle curr, prev, sub, data, varD, pos, varP, encP, err, sensId, errId, idList\rangle$$

where

$curr = (Enc^*(n, k(n, TP)), Enc^*(q + 1, k(n, TP)))$,
$prev = \epsilon$[4],
$sub = (Enc^*(n, k(n, TP)), Enc^*(q + 1, k(n, TP))$,
$data = Enc(d, k(n, SS))$,
$varD = 0$,
$pos = (x_n, y_n)$,
$varP = 0$
$encP = (Enc(x_n, k(n, TP)), (Enc(y_n, k(n, TP))))$[5]
$err = 0$,
$sensId = \epsilon$,
$errId = \epsilon$,
$idList = [(Enc^*(n, k(n, TP)), Enc^*(q + 1, k(n, TP)))]$

Once ready $m_{n,q+1}$ is copied into the local table[6] and then it is queued in the transmission buffer.

[4] ϵ denotes an empty field.

[5] Since coordinates have a finite representation, we assume, for the sake of simplicity, that they are represented by integer numbers

[6] Only some of the fields are actually stored as explained at the beginning of this section

4.2.2. Message Reception and Integrity Verification

Let n be a node receiving a message $m_{j,h}$, and let q be the number of messages already transmitted by n over the network. The message is analyzed to find out whether it was originally transmitted by the node itself. This can be done searching the local table using the content of field *prev* as hash key. If the search succeeds, then $m_{j,h}$ was transmitted by node n and therefore it can verify the integrity of the received message. Instead, if the search fails node n has to re-transmit the message over the network. Notice that in the former case n acts as a *Notifier-Controller* (NC), while in the latter it acts as a *Transmitter-Processor* (TP).

1 *Forwarding messages*

A new message $m_{n,q+1}$ is therefore prepared by setting field *curr* to

$(Enc^*(n, k(n, TP)), Enc^*(q + 1, k(n, TP)))$

and field *prev* to the value of field *curr* of the received message. All the others fields are set to the value of the corresponding field in the received message. Once ready $m_{n,q+1}$ is copied into the local table and then it is queued in the transmission buffer.

2 *Verifying the integrity*

The node verifies the integrity of the received message by comparing it with the information retrieved from its local table. This is done by checking whether fields *data* and *encP* of the received message match with the corresponding fields retrieved from the local table. If they do then message integrity is preserved and therefore the received message can be dropped. Instead, if they do not then the received message should be considered as corrupted and therefore the node transmits over the network an error message structured as follows:

$m_{n,q+1} = \langle curr, prev, sub, data, varD, pos, varP, encP, err, sensId, errId, idList\rangle$

where

$curr = (Enc^*(n, k(n, TP)), Enc^*(q + 1, k(n, TP))),$

$prev = \epsilon,$

$sub = (Enc^*(n, k(n, TP)), Enc^*(q + 1, k(n, TP))),$

$data = Enc(LocTab.data, k(n, NC)),$

$varD = 0,$

$pos = (x_n, y_n),$

$varP = 0,$

$encP = Enc(LocTab.encP, k(n, NC))$[7],

[7]Function Enc is applied to both values of field *encP* resulting in a new pair of encrypted values.

$err = 1,$

$sensId = LocTab.sub,$

$errId = m_{j,h}.curr,$

$idList = m_{j,h}.idList$

where $m_{j,h}.f$ denotes the field f of message $m_{j,h}$ and $LocTab.f$ denotes the field f retrieved from the local table.

Notice that field *prev* is left empty to avoid loops with the malicious node and/or the spreading of different and opposite error messages; field *sub* identifies the node that found the error (i.e., node n); field *sensId* equals the content of field *sub* retrieved from the local table and therefore provides information on which node originally sensed or aggregated the datum; field *errId* equals field *curr* of the received message to provide information on where the error occurred; *error* is set to 1 to indicate that the current message is an error message. The fields *data* and *encP* equal the corresponding fields retrieved from the local table and are encrypted with the *Notifier-Controller* key $k(n, NC)$. Finally, the new message is queued in the transmission buffer once it has been copied in the local table.

4.2.3. Data Aggregation

Whenever the number of messages in the transmission buffer exceeds a given threshold messages are aggregated in order to avoid buffer overflow. In what follows we discuss the way in which messages to be aggregated are selected, how the aggregation process works and finally how a new aggregated message is structured.

1 *Messages Selection.*

The messages selection procedure iteratively operates to arrange enqueued messages in a suitable number of aggregation groups. Then the messages of each aggregation group are merged into a single message. As a result the number of messages in the transmission buffer decreases. Notice that error messages (i.e., messages having the field *err* set to 1) are not considered.

The goal is to decrease the length of the transmission queue, at a given instant tj of $|uV(tj)|$ messages (see Section 3.1). Since at the end of the aggregation process A new aggregated messages will be inserted in the queue, in order to guarantee that at the end of the aggregation process the queue length decreases of $|uV|$ messages it is necessary to remove $|uV| + A$ messages. The value A is not predictable in advance, but it is computed using the following iterative procedure.

For each message m stored in the transmission buffer, let $m.group$ be a variable whose initial value is 0. Let A denote the number of aggregation groups found, whose initial value equals 1. Finally, let *dynamicList* be a list of messages initially empty.

During the first iteration, we look for messages generated by nodes whose position falls within a circular area of radius R (which is a parameter of the DyDAP algorithm) whose center is given by the position of the first message in the buffer (see Fig. 5). All the found messages are marked as belonging to the first aggregation group by setting to 1 their variable *group*. Furthermore, all these messages are put in *dynamicList*. Let *num*

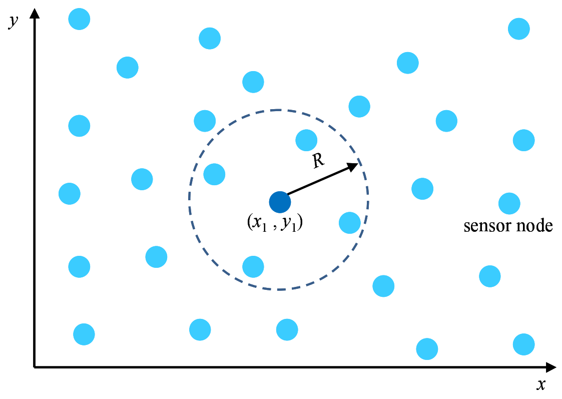
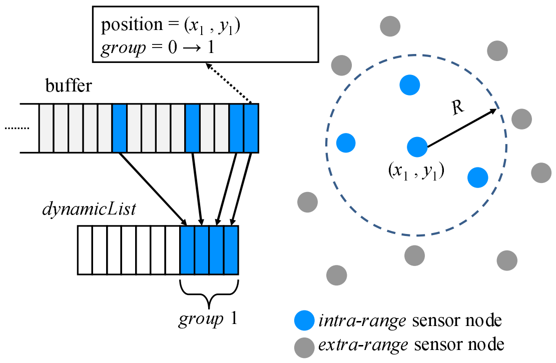


Figure 5: Message searching range.

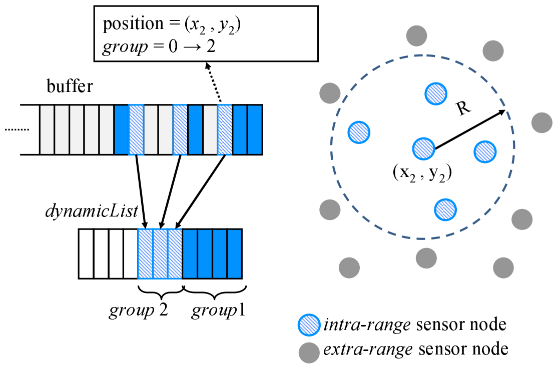
be the number of messages in *dynamicList*, that is the number of messages that can be removed from the transmission buffer. If *num* is smaller than $|u_V| + A$ (i.e., it is not enough to remove from the queue *num* messages), a new aggregation group is created. This is done by incrementing A and iterating the procedure. Thus, during the next iteration, the procedure looks for the first message in the queue that still has *group* equal to 0. Then it sets *group* to A and uses the value in field *pos* as the center of the new area of radius R in which looking for messages to be aggregated. To avoid that the same message is aggregated into more than one group, only messages having *group* equal to 0 are taken into account.

This is repeated until the number of messages in *dynamicList* is equal to $|u_V| + A$ or all the messages have been assigned to groups. When this condition is satisfied the procedure terminates and messages are aggregated as described in the following step.

Figure 6 shows how the messages selection procedure works. In the example, 5 messages have to be removed from the queue, i.e., $u_V = -5$. At the end of the first iteration, $A = 1$ and the number of messages with *group* equal to 1 is 4, that is the first aggregration group is made up of 4 messages. Moreover, *dynamicList* contains such messages and since they are less than $|u_V| + A$, the procedure is iterated with A incremented by 1. At the end of the second iteration 3 messages have been selected with variable *group* equal to 2. Thus, *dynamicList* contains 3 more messages, i.e., the number of messages that are strictly required to reach the target $|u_V| + A$.


position = (x_1, y_1)
group = $0 \rightarrow 1$
buffer
dynamicList
group 1
R
(x_1, y_1)
● *intra-range* sensor node
● *extra-range* sensor node

(a) First Iteration


position = (x_2, y_2)
group = $0 \rightarrow 2$
buffer
dynamicList
group 2
*group*1
R
(x_2, y_2)
▨ *intra-range* sensor node
● *extra-range* sensor node

(b) Second Iteration

Figure 6: Steps of the message searching procedure.

2 *Data Aggregation.*

Messages in *dynamicList*, having the same *group* value, are joined together and therefore a new message is generated for each aggregation group.

Let be G an aggregation group composed of J different messages. The fields *data* and *pos* of the aggregated message are computed as a weighted encrypted sum of the corresponding values in the J messages in G. In particular, the weight of the $i-th$ message ($w_i, i = 1..J$) equals the size of the field *idList*. As a consequence, the weight w of a message resulting from the aggregation of J different messages is equal to the sum of the weights of each joined message, that is:

$$w = \sum_{i=1}^{J} w_i \tag{7}$$

According to equation (5) the following expressions hold for fields *data*, *pos* and *encP*:

$$\begin{aligned} data &= \sum_{i=1}^{J} w_i m_i.data, \qquad (8) \\ pos &= \left(\frac{1}{w} \sum_{i=1}^{J} w_i m_i.pos.x, \frac{1}{w} \sum_{i=1}^{J} w_i m_i.pos.y \right), \qquad (9) \\ encP &= \left(\frac{1}{w} \sum_{i=1}^{J} w_i m_i.encP.x, \frac{1}{w} \sum_{i=1}^{J} w_i m_i.encP.y \right) \qquad (10) \end{aligned}$$

where, m_i denotes the i-th message in the aggregation list.

Furthermore, the variances of data and position of the aggregated messages can be computed. In particular, considering equation (8), the variance of the data equals [26]:

$$varD = \frac{1}{w-1} \sum_{i=1}^{J} \left[(w_i - 1)\sigma_{d_i}^2 + w_i d_i^2 \right] - \frac{w}{w-1} data^2 \tag{11}$$

where d_i and σ_{d_i} denote the value of fields *data* and *varD* of the i-th message in the aggregation list. Similarly, it is possible also to compute *varP*. The values of the variance measure the accuracy of the information resulting from the aggregation.

3 *Message Structure.*

The new message $m_{n,q+1}$ generated by node n, assuming that it has already transmitted q messages over the network, is structured as follows;

$$m_{n,q+1} = \langle curr, prev, sub, data, varD, pos, varP, encP, err, sensId, errId, idList \rangle$$

where

$curr = (Enc * (n, k(n,TP)), Enc * (q+1, k(n,TP)))$,

$prev = \epsilon$,

$sub = (Enc * (n, k(n,TP)), Enc * (q+1, k(n,TP)))$,

$data =$ see eq. 8,

$varD =$ see eq. 11,

$pos =$ see eq. 9,

$varP =$ as in eq. 11,

$encP =$ see eq. 10,

$err = 0$,

$sensId = \epsilon$,

$errId = \epsilon$,

$idList = \bigcup_{i=i}^{J} m_i.idList$

Notice that fields *curr* and *sub* show which node carried out the aggregation process, while field *prev* is empty since this is a new message. Fields *varD*, *pos*, *data* and *varP* are set according to equations (8) - (11). Finally, field *idList* is obtained by merging the corresponding fields of the J messages involved in the aggregation process.

Once built the message is copied in the local table before being enqueued in the transmission buffer.

4 *Dealing with an upper bound on packet size.*

So far, the description of the aggregation algorithm has been outlined under the implicit assumption that the size of aggregated packets is smaller than the maximum allowed one. We recall that the size of aggregated packets depends on the number of entries in *idList*. It is easy to take into account an upper bound on packet size by simply stopping the creation of an aggregation group as soon as the total number of IDs exceeds id^M, which is the maximum number of IDs that can be stored in the list.

4.3. Towards policy enforcement and trust

Privacy policy enforcement consists in verifying the compliance of the actions performed by nodes with a given privacy policy. In general, there are two ways in which such a verification can be carried out: the first way consists in providing *ex-post* enforcement mechanisms, that is the controls are done after all the actions related to a policy are performed (e.g., audit-based mechanisms). The second one consists in having *run-time* enforcement mechanisms that is, the effect of every action is checked before actual execution.

The *Message Reception and Integrity Verification Protocol* implements an *ex-post* mechanism since the node checks the integrity of the received message with the obligation of sending an error message to the sink in case a corrupted message is detected.

Instead, *run-time* enforcements could be implemented by adding trust management mechanisms. In such a case a new key has to be used to assure that communication takes place only among authorized nodes belonging to the network. In fact, a further requirement concerning (communication) trust states that only authorized nodes are allowed to communicate within the system. This requirement can be satisfied by introducing an *Authenticator-Processor* (AP) key known by every node of the network. As a consequence a node can encrypt messages with the AP key, assuring in this way that the transmitter is a trusted node of the network. The main drawback consists in the fact that a node receiving a message needs to decrypt it (using the AP key) before any processing can occur. Notice that this mechanism prevents external untrusted nodes from both accessing messages that are traveling across the network and transmitting malicious messages over the network.

Although the above technique can improve the level of trust among the nodes of the network, it requires relevant computational and power resources, since the number of encryptions/decryptions is equal to the number of message transmissions and receptions. Hence, this solution could be applied to the next generation of sensor technologies [27, 28] and to hybrid architectures, such as those composed of mesh networks and WSN [29].

5. Performance evaluation

This section discusses the effectiveness of the DyDAP approach with respect to data accuracy and end-to-end secure data aggregation by showing the results of several simulations. In particular, we analyze transmission buffer overflow, estimation accuracy, and energy consumption.

Notice that the features of DyDAP concerning privacy such as anonymity and data integrity are not the target of such simulations even though they are achieved by means of the protocols investigated herein. In fact, privacy management depends on the design choices made in order to develop DyDAP according to the privacy model described in Section 2.

As a result only data accuracy and end-to-end secure data aggregation are tested by means of computer simulations that reproduce a real application context. The results show that DyDAP is able to exploit end-to-end secure aggregation to face network congestion while providing, at the same time, a high sensing accuracy. Finally, it has been also shown that DyDAP exhibits a negligible sensitivity with respect to the parameters q_T, K, and T of the control algorithm.

5.1. The example

The chosen example is a temperature monitoring problem based on real data traces, obtained from the 1968 Lone Pine Canyon (CA, USA) wildfire [30]. We considered this use-case in order to test DyDAP in a real-world strenuous scenario with large temperature excursions. This can be considered a worst case study; in fact, when the monitored field exhibits strong variations, estimation errors due to network congestion can severely impair WSN monitoring accuracy. The simulations were carried out using the Castalia simulator, v.1.3 [31], considering a sensors field having a 500m x 100m rectangular shape has shown in Fig. 7 and assuming that sensor nodes are equipped with the Texas Instruments (http://www.ti.com) radio transceiver CC2420. The default Castalia gradient-based routing algorithm has been used [59], setting up a tree topology having the sink as root node. The values of the parameters used for the simulations

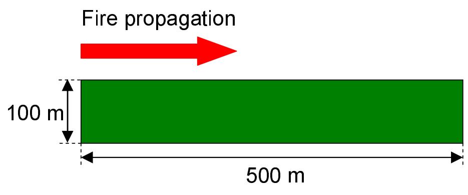


Figure 7: Sensor field.

are reported in Tab. 1.

Table 1: Parameters for simulations

Parameter	*Description*	*Value*
N	Number of nodes	$[56, 500]$
A	Sensor Field size	500×100 m^2
C_B	Buffer capacity	100 messages
t_P	Acquisition Period	1; 1.5; 2 s
q_T	Target queue level	70%-90% C_B
K	Constant of PI controller	$\{0.5, 1\}$
T	Constant of PI controller	$\{1, 4, 8\}$
R	Data Aggregation Range	30 m
t_n	Sampling time of DyDAP	500 ms
t_S	Duration of simulation	240 s

In order to exploit the header compression gain due to 6LoWPAN standard, we have encapsulated DyDAP messages in a IPv6 over IEEE 802.15.4 stack as specified in standards [57] and [58]. Notice that according to such standards the maximum packet size equals 127 bytes. However, during all simulations the actual packet size did not exceed such a maximum even when data aggregation was carried out.

5.2. Parameter sensitivity of DyDAP

To investigate the parameter sensitivity of DyDAP, we have evaluated key performance indexes such as the relative estimation error, the ratio of lost packets, and the number of transmitted packets by varying q_T, K and T parameters according to Tab. 1. Results show negligible variations of the WSN behavior with respect to the parameter sets. In particular, Figs. 8 and 9 report the mean value of the absolute relative error in temperature estimation and the overall number of messages transmitted in the WSN, respectively. Values are obtained using the DyDAP algorithm for $t_P = 1$ s[8] In Figs. 8 and 9, we refer to each parameter set by using a label Si on the x-axes where i refers to the i-th set of Tab. 2.

Table 2: Parameter set labels

Set	q_T	K	T
S1	70% C_B	0.5	1
S2	70% C_B	0.5	4
S3	70% C_B	0.5	8
S4	70% C_B	1	1
S5	70% C_B	1	4
S6	70% C_B	1	8
S7	80% C_B	0.5	1
S8	80% C_B	0.5	4
S9	80% C_B	0.5	8
S10	80% C_B	1	1
S11	80% C_B	1	4
S12	80% C_B	1	8
S13	90% C_B	0.5	1
S14	90% C_B	0.5	4
S15	90% C_B	0.5	8
S16	90% C_B	1	1
S17	90% C_B	1	4
S18	90% C_B	1	8

5.3. Simulations results

In what follows, unless otherwise specified, we refer to results obtained using parameter set S10 without loss of generality. The first simulation, whose results are reported in Fig. 10, shows the dynamic evolution of the queue of a sensor node in a simulated network with $N = 300$, and $t_P = 1$ s. It can be seen that the use of the proposed congestion control algorithm, based on data aggregation, avoids buffer saturation. In fact, the queue level is always around the target queue level $q_T = 80\%$ of buffer capacity C_B. Instead, when DyDAP is not used, the transmission buffer very often overflows, with a consequent increase of the message loss rate. This is clearly shown also in Fig. 11 which reports the ratio of lost packets when DyDAP is not used.

[8]Results obtained for other values of t_P have not been shown because very similar to those already reported here.

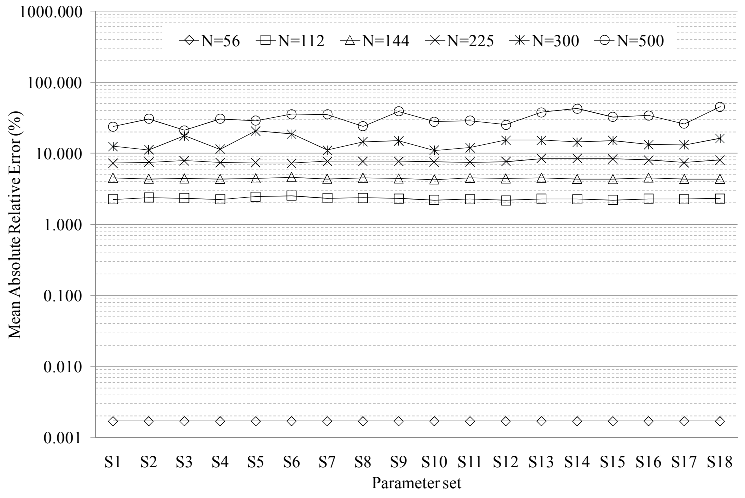


Figure 8: Mean Absolute Relative of DyDAP for several parameter sets ($t_P = 1$ s).

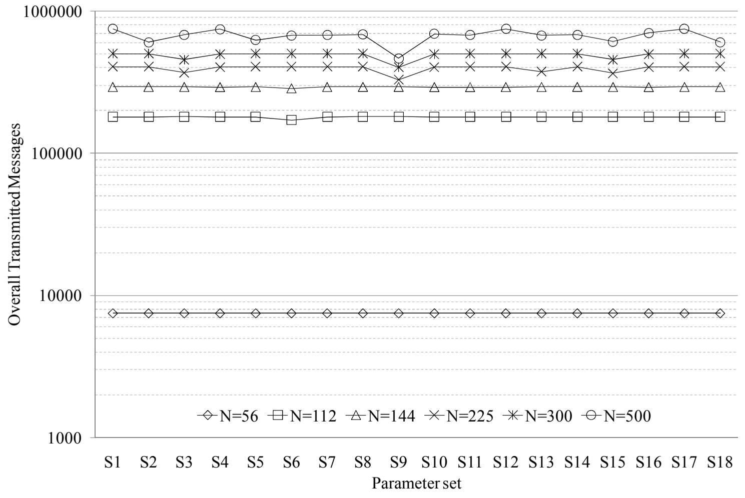


Figure 9: Overall number of transmitted messages using DyDAP for several parameter sets ($t_P = 1$ s).

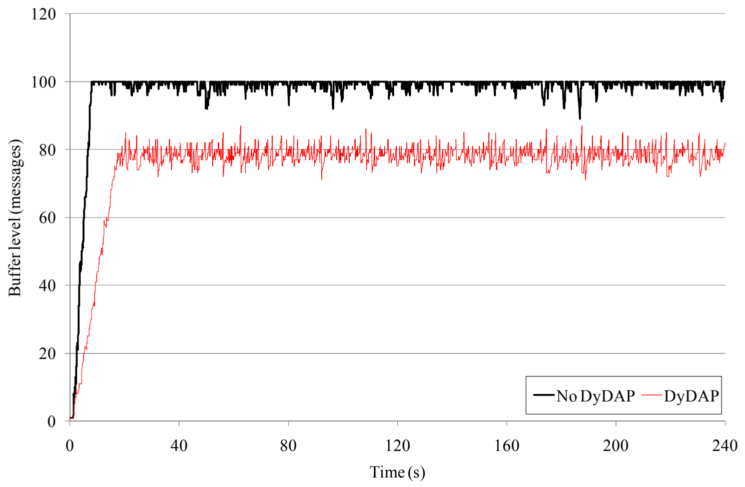


Figure 10: Dynamic evolution of a queue with and without DyDAP algorithm.

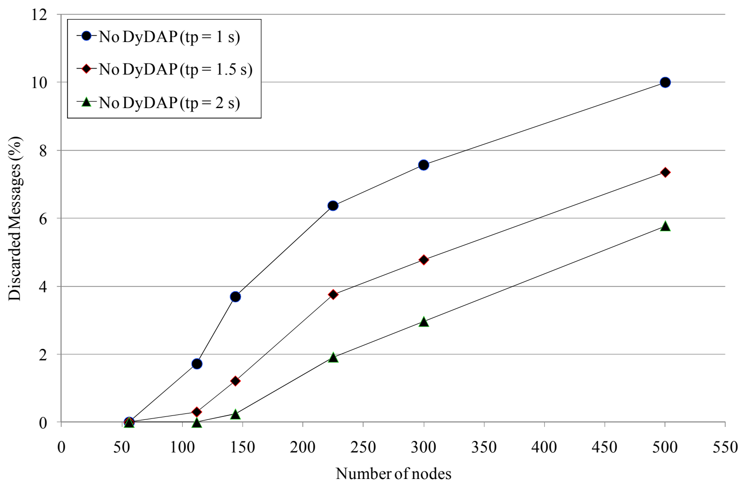


Figure 11: Percentage of lost packets when DyDAP is not used.

To evaluate the improvement in the estimation accuracy caused by the reduction of the loss rate, we have compared the actual data of the temperature field with those estimated by the sink when DyDAP is turned on and off, respectively. Fig. 12 reports the mean and maximum values for the absolute relative error of the temperature estimations (i.e., the absolute value of the ratio among the temperature estimation error and the real temperature value)[9]. The simulations have been carried out using different values for t_P (1s, 1.5s and 2s). The results show that accuracy decreases by increasing the number of nodes because of network congestion and that DyDAP data aggregation provides a great benefit in term of accuracy. To provide a further insight on this very important issue, Fig. 13 reports the mean of the absolute relative estimation error, registered at each sensor node position, in the case of $N = 225$ and $t_P = 1$s. The color of each node represents the mean of the absolute relative estimation error measured in the node position. The error is represented in a gray scale: black (white) corresponds to 100% (0%). This provides information about the spatial distribution of the estimation error showing that network congestion can severely impair the WSN behavior. In fact, without using DyDAP data aggregation almost half of the sensor nodes provides an estimation error very close to 100% due to lost messages (dark nodes in Fig. 13.a).

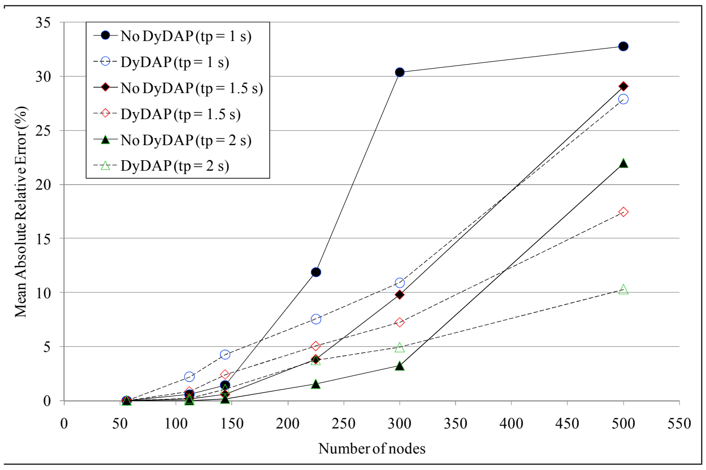
In the considered fire detection scenario it is also important to show how the estimated and real temperatures differ over time: Fig. 14 and 15 report the actual temperature values and the ones estimated by the sink using aggregated data: the gain in estimation accuracy is very impressive. For example, for Node 0, without using DyDAP data aggregation, (see Fig. 14.a) the temperature peak is fully missed because the network congestion does not allow messages to be delivered to the sink. Thus, without using DyDAP the estimation accuracy is so bad that the entire WSN misses its sensing target.

Finally, to evaluate energy consumption we analyzed the number of messages that each node in the network transmits (retransmissions included) since transmission is the most energy consuming task of a sensor node. Fig. 16 reports the number of transmitted messages, in different operative conditions, showing that using DyDAP a slightly smaller number of messages is transmitted. In conclusion, DyDAP avoids network congestion and therefore greatly improves estimation accuracy, without increasing in a significant way the amount of energy required.

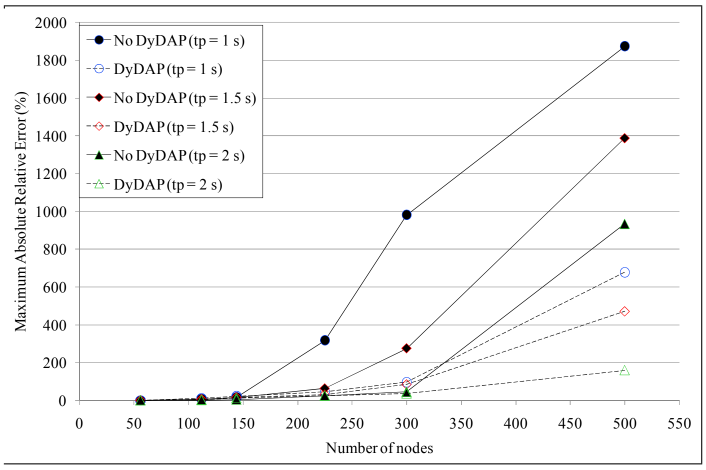
6. Related works

WSN applications require to collect a large amount of data and, due to the limited resources in terms of power of sensor nodes, it is necessary to aggregate such data in order to reduce the amount of transmitted information. The

[9]In these diagrams we have averaged, over all sensor nodes, the mean (or the maximum) of the absolute relative estimation errors.

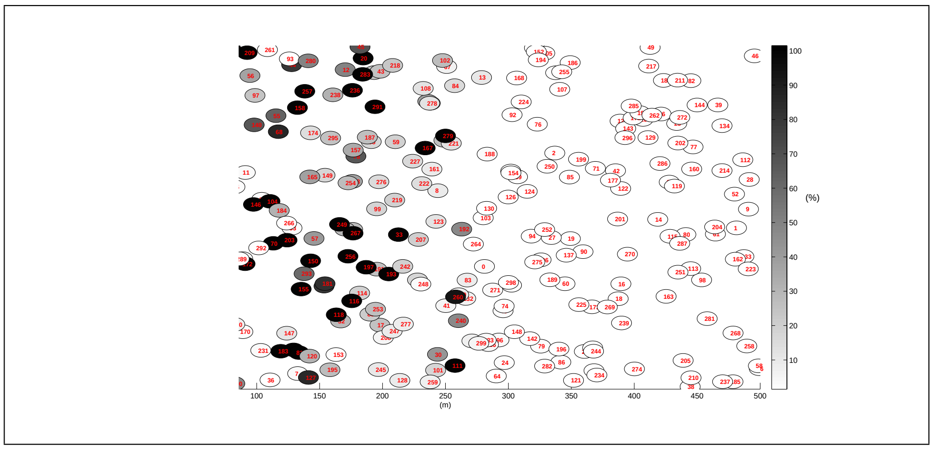


(a) Mean error



(b) Maximum error

Figure 12: Absolute Relative Error between the actual sensed values from the sensor nodes and the received values from the sink node.


(a) without DyDAP

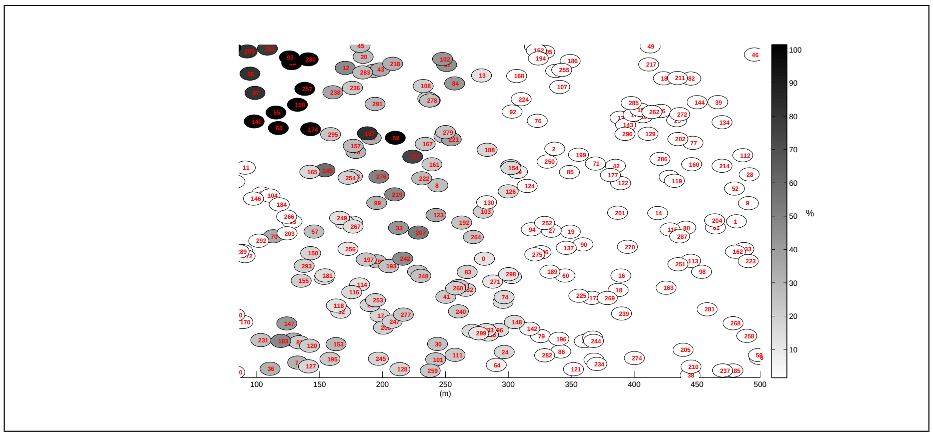
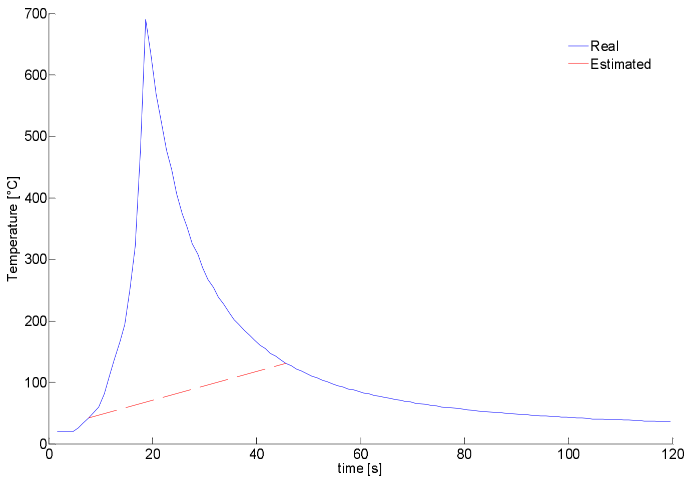
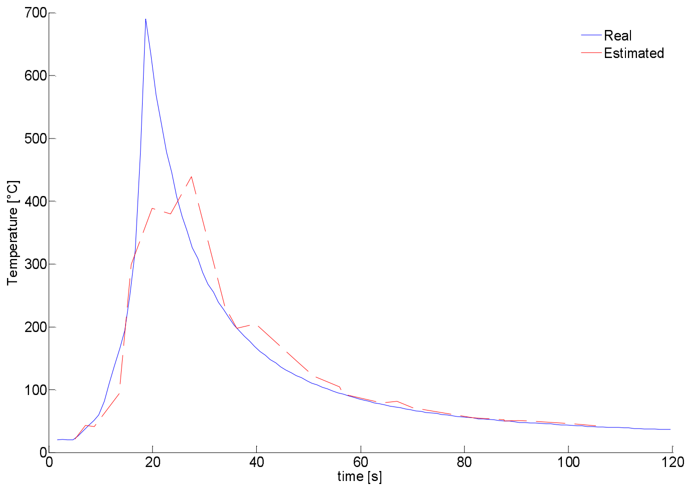

(b) with DyDAP

Figure 13: Spatial distribution of the average relative estimation error ($N = 300$ and $t_p = 1$s).

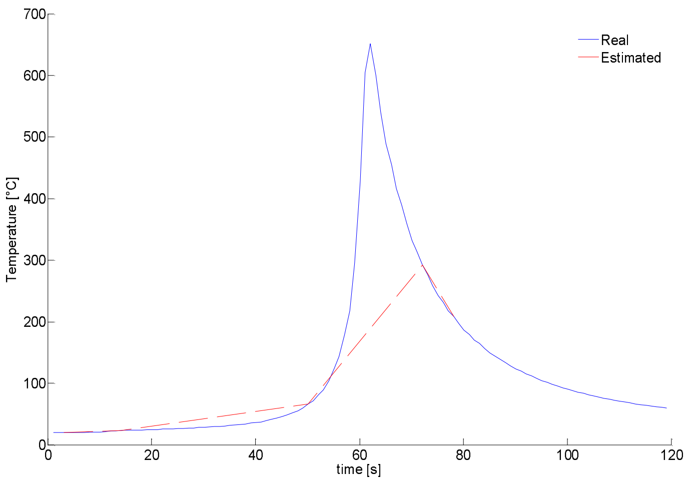


(a) Node 57 without DyDAP

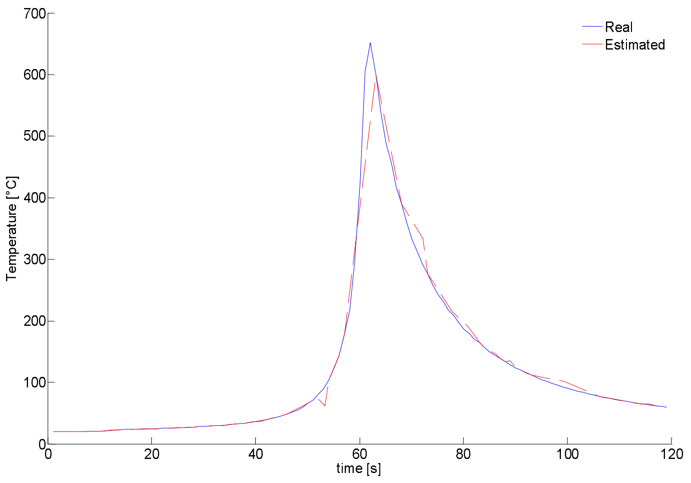


(b) Node 57 with DyDAP

Figure 14: Relative Error between the actual sensed values from the sensor node and the received value from the sink node ($N = 225$, $t_p = 1$ s).



(a) Node 227 without DyDAP



(b) Node 227 with DyDAP

Figure 15: Relative Error between the actual sensed values from the sensor node and the received value from the sink node ($N = 300$, $t_p = 1$ s).

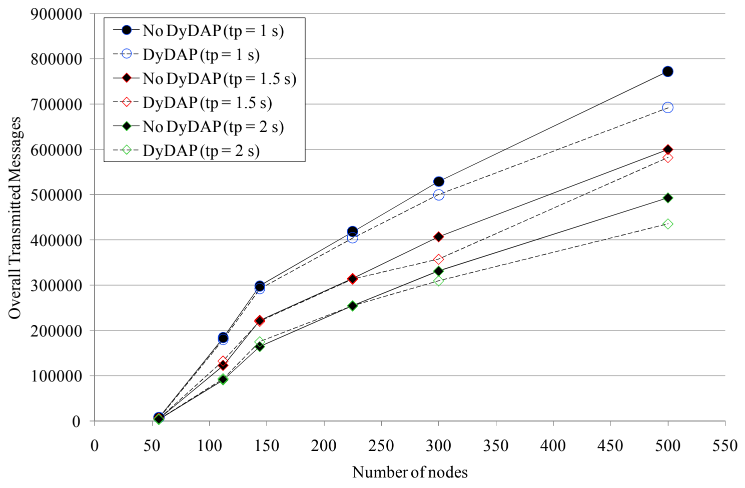


Figure 16: Total number of transmitted messages in a network with N nodes.

wireless nature of the communication channel along with the remote access increase the risk of attacks that can lead to violations of privacy, integrity and confidentiality. In particular the following common threats may occur [32, 33]:

- Eavesdropping: a malicious user could easily discover the communication content by listening to the communication occurring among nodes.
- Masking: a malicious node may mask its real nature behind the identity of a node that is authorized to take part to the communication in order to misroute messages.

In order to preserve privacy in WSN, the available solutions may be classified into two main groups: anonymity mechanisms based on data cloaking [32, 34] and privacy policy based approaches [35].

The former approach is based on perturbing data following some kind of criterion, for instance K-anonymity guarantees that every record is indistinguishable from at least k-1 other records [36]. In [32, 34, 37, 38] four main data cloaking anonymity approaches ([32, 37] specific for cloaking localization information) are proposed:

- Decentralize Sensible Data: the basic idea of this approach is to distribute the sensed location data through a spanning tree, so that no single node holds the complete view of the original data.
- Secure Communication Channel: the use of a secure communication protocols, such as SPINS [39], reduces the eavesdropping and active attack risk by means of encryption techniques.
- Change Data Traffic: the traffic pattern is altered with some bogus data that obfuscate the real position of the nodes.

- Node Mobility: the basic idea is to move the sensor nodes in order to change dynamically the localization information, making it difficult to identify the node.

For instance, [32] proposes a solution that guarantees the anonymous usage of location based information. More specifically, such a solution consists of a cloaking algorithm that regulates the granularity of location information to meet the specified anonymity constraints. This work focuses on localization services and therefore constrains the middleware architecture needed to support the proposed algorithm. Hence, such a solution cannot be considered a general context independent anonymity approach.

Privacy policy based approaches [35, 40, 41] state who can use individuals data, which data can be collected, for what purpose the data can be used, and how they can be distributed. A common policy based approach addresses privacy concerns at database layer after data have been collected [41]. Other works [42] address issues such as access control and authentication. Duri et al. [35] propose a policy-based framework for protecting sensor information. The Mist routing project for mobile users [43] combines location privacy with communication aspects. It faces the source location privacy problem by designing *ad hoc* routing protocol that keeps the location private from source to routers.

Our work provides a contribution in the field of privacy by defining a role-based context-independent solution that guarantees nodes anonymity before data are collected into a database. Thus, our solution may be combined with both data cloaking mechanisms and some other privacy policy based approaches.

Secure data aggregation in WSN is a very mature research field and the literature reports many solutions addressing at the same time aggregation issues and security aspects such as confidentiality, integrity, authentication, and availability (an exhaustive and very comprehensive view of this topic can be found in [12]). The approaches proposed so far can be classified into two big families depending on whether the hop-by-hop or end-to-end cryptography is used. Hop-by-hop encryption is usually based on symmetric key schemes, which demand less computing resources than asymmetric key ones. These algorithms, such as [4, 5, 44, 45, 46, 47, 48, 49, 50], require each aggregator to decrypt every message it receives to allow in-network processing, thus causing a confidentiality breach. Furthermore, applying several consecutive encryption/decryption operations can negatively impair latencies. Finally, hop-by-hop aggregation requires each node to share secret keys with all its neighbors. In order to face these problems, aggregation algorithms able to work on ciphered data, using either asymmetric or symmetric keys [51]-[56] have been proposed. However, the main limitation of such approaches is that they allow very simple aggregation functions to be used, such as sum and average [12]. Despite this very broad variety of proposals, no single solution has been conceived yet to address confidentiality, integrity, adaptive aggregation and privacy issues at the same time, as DyDAP does.

7. Conclusion

This paper has presented the problem of secure end-to-end data aggregation and privacy management in WSN. An innovative integrated solution for dynamic data aggregation with privacy function, DyDAP, has been presented. The elements involved in the management of privacy-related information are represented using UML modeling, whereas an algorithm based on discrete-time control theory is exploited for aggregating data. The effectiveness of the proposed framework has been verified using computer simulations. The results show that DyDAPis able to reduce the network load in case of congestion and to improve WSN estimation accuracy, while, at the same time, it guarantees anonymity management and data integrity.

At the moment we are planning to extend our work along several directions in order to analyze in depth the DyDAP behavior and to improve its efficiency. In particular, we are experimenting the application of DyDAP to multimedia sensor networks whose nodes may exchange audio and video signals, showing how the proposed solution can be tailored to many application domains. Furthermore, we are evaluating lightweight optimization strategies for minimizing the variance of the aggregated data, also considering a broader set of topological configurations, such as those containing multiple sinks. In order to improve the level of security towards attacks we are studying game-theoretic approaches to model malicious nodes. As regard data trust we shall combine DyDAP with other cross layer information, i.e., information dealing with node reputation information. Moreover, in this direction we also envisage the possibility of using emerging nano-technologies or hybrid architectures to overcome the power limits. Finally, we will evaluate DyDAP as building block of several secure Internet of Things (IoT) scenarios.

References

[1] I. F. Akyildiz, T. Melodia, K. Chowdhury, A survey on wireless multimedia sensor networks, Elsevier Computer Networks Journal.

[2] D. Dardari, A. Conti, C. Buratti, R. Verdone, Mathematical evaluation of environmental monitoring estimation error through energy-efficient wireless sensor networks, IEEE Transactions on Mobile Computing 6 (7) (2007) 790–802.

[3] O. B. Akan, I. F. Akyildiz, Event-to-sink reliable transport in wireless sensor networks, IEEE/ACM Transactions on Networking 13 (5) (2005) 1003–1016.

[4] M.Bagaa, N.Lasla, A. Ouadjaout, Y. Challal, Sedan: Secure and efficient protocol for data aggregation in wireless sensor networks, in: In Proceedings of IEEE LCN, Dublin, Ireland, 2007.

[5] A. Mahimkar, T. Rappaport, Securedav: a secure data aggregation and verification protocol for wireless sensor networks, in: 47th IEEE Global Telecommunications Conference (Globecom), 2004.

[6] C. Castelluccia, E. Mykletun, G. Tsudik, Efficient aggregation of encrypted data in wireless sensor networks, in: Conference on Mobile and Ubiquitous Systems: Networking and Services, 2005.

[7] L. Hu, D. Evans, Secure aggregation for wireless networks, in: Workshop on Security and Assurance in Ad Hoc Networks, 2003.

[8] O. Younis, M. Krunz, S. Ramasubramanian, Node clustering in wireless sensor networks: recent developments and deployment challenges, Network, IEEE 20 (3) (2006) 20–25.

[9] E.Fasolo, M. Rossi, J. Widmer, M. Zorzi, In-network aggregation techniques for wireless sensor networks: A survey, IEEE Wireless Communications.

[10] T. Mastrocristino, G. Tesoriere, L. A. Grieco, G. Boggia, M. R. Palattella, P. Camarda, Control based on data-aggregation for wireless sensor networks, in: Proc. of IEEE Int. Symp. on Industrial Electronics, ISIE2010, Bari, Italy, 2010.

[11] L. A. Grieco, G. Boggia, S. Sicari, P. Colombo, Secure wireless multimedia sensor networks: a survey, in: Proc. of The Third Int. Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM, Sliema, Malta, 2009.

[12] S. Ozdemir, Y. Xiao, Secure data aggregation in wireless sensor networks: a comprehensive overview, Computer Networks 53.

[13] A. Coen-Porisini, P. Colombo, S. Sicari, A. Trombetta, A conceptual model for privacy policies, in: Proc. of SEA 2007, Cambridge (MS), USA, 2007.

[14] A.Coen-Porisini, P.Colombo, S.Sicari, Dealing with anonymity in wireless sensor networks, in: Proc. of 25th annual ACM symposium on Applied Computing (ACM SAC), Sierre, Switzerland, 2010.

[15] A.Coen-Porisini, P.Colombo, S.Sicari, ”privacy aware systems: from models to patterns”, in: Software Engineering for Secure Systems: Industrial and Research Perspectives, IGI Global, 2010.

[16] OMG, Unified Modeling Language: Infrastructure, Ver. 2.1.2, formal/2007-11-02 (November 2007).

[17] OMG, Unified Modeling Language: Superstructure, Ver. 2.1.2, formal/2007-11-02 (November 2007).

[18] K. J. Astrom, B. Wittenmark, Computer controlled systems: theory and design, 3rd Edition, Prentice Hall, Englewood Cliffs, 1995.

[19] Directive 95/46/EC of the European Parliament. Official Journal of the European Communities of 23 November 1995 No L. 281 p. 31.

[20] Q.Ni, A.Trombetta, E. Bertino, J. Lobo, Privacy-aware role based access control, in: Proceedings of the 12th ACM symposium on Access control models and technologies, ACM, New York, NY, USA, 2007.

[21] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, Y. F. Hu, Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards, Computer Communications 30 (7) (2007) 1655–1695.

[22] B. H. Walke, S. Mangold, L. Berlemann, IEEE 802 Wireless Systems, John Wiley & Sons, Ltd, NJ, USA, 2006.

[23] H. Zhanga, A. Arorab, Y. Choic, M. Goudac, Reliable bursty convergecast in wireless sensor networks, Elsevier Computer Communications 30 (13) (2007) 2560–2576.

[24] V. Jacobson, Congestion avoidance and control, SIGCOMM Comput. Commun. Rev. 18 (4) (1988) 314–329.

[25] G. Boggia, P. Camarda, L. A. Grieco, S. Mascolo, Feedback-based control for providing real-time services with the 802.11e MAC, IEEE/ACM Trans. on Networking 15 (2) (2007) 323–333.

[26] A. Papoulis, Probability, Random Variables and Stochastic Processes, 3rd Edition, Mc Graw Hill, 1991.

[27] I. F. Akyildiz, F. Brunetti, C. Blazquez, Nanonetworking: A new communication paradigm, Elsevier Computer Networks Journal.

[28] J. P. M. She, J. T. W. Yeow, Nanotechnology-enabled wireless sensor networks: From a device perspective, IEEE Sensors Journal 6 (5).

[29] S. Sicari, R. Riggio, Secure aggregation in hybrid mesh and wireless sensor networks, in: Proc. of IEEE ICUMT'09, St. Petersburg, Russia, 2009.

[30] A. DeBano, F. leonard, M. Rice, C. Conrad, Soil heating in chaparral fires: effects on soil properties, plant nutrients, erosion, and runoff, Pacific Southwest Forest and Range Experiment Station P.O. Box 245, Res. Paper PSW-145.

[31] Castalia simulator - official website, http://castalia.npc.nicta.com.au.

[32] M. Gruteser, G. Schelle, A. Jain, R. Han, D. Grunwald, Privacy-aware location sensor networks, in: In Proceedings of the 9th USENIX Workshop on Hot Topics in Operating Systems (HotOS IX), 2003.

[33] H. Chan, A. Perrig, Security and privacy in sensor networks, IEEE Computer Magazine (2003) 103–105.

[34] N. B. Priyantha, A. Chakraborty, H. Balakrishnan, The cricket location support system, in: In Proceedings of ACM Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM), 2000.

[35] M. G. S. Duri, P. M. X. Liu, R. Perez, M. Singh, J. Tang, Framework for security and privacy in automotive telematics, in: In Proceedings of 2nd ACM International Worksphop on Mobile Commerce, 2000.

[36] P. Samarati, L. Sweeney, Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression, Technical Report SRI-CSL-98-04, Computer Science Laboratory, SRI International.

[37] M. Gruteser, D. Grunwald, A methodological assessment of location privacy risks in wireless hotspot networks, in: In Proceedings of the first International Conference on Security in Pervasive Computing, 2003.

[38] A. Smailagic, D. P. Siewiorek, J. Anhalt, Y. W. D. Kogan, Location sensing and privacy in a context aware computing environment, in: In Proceedings of Pervasive Computing, 2001.

[39] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, D. E. Culler, Spins: security protocols for sensor networks, Wireless Networking 8 (5) (2002) 521–534.

[40] M. Langheinrich, A privacy awareness system for ubiquitous computing environments, in: In Proceedings of the 4th Int. Conf. on Ubiquitous Computing, 2002.

[41] E. Snekkenes, Concepts for personal location privacy policies, in: In Proceedings of 3rd ACM Conf. on Electronic Commerce, 2001.

[42] D. Molnar, D. Wagner, Privacy and security in library rfid : Issues, practices, and architectures, in: In Proceedings of ACM CCS, 2004.

[43] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. D. Mickunas, S. Yi, Routing through the mist: privacy preserving communication in ubiquitous computing environments, in: In Proceedings of IEEE Int. Conf. on Distributed Computing systems (ICDS), Vienna, Austria, 2002.

[44] B. Przydatek, D. Song, A. Perrig, Sia : secure information aggregation in sensor networks, in: SenSys'03, 2003.

[45] H. Cam, S. Ozdemir, P. Nair, D. Muthuavinashiappan, H. Sanli, Energy-efficient and secure pattern based data aggregation for wireless sensor networks, Comput. Commun., Elsevier 29 (4) (2006) 446—455.

[46] W. Du, J. Deng, Y. Han, P. Varshney, A witness-based approach for data fusion assurance in wireless sensor networks, in: IEEE Global Telecommunications Conference (GLOBECOM '03), 2003.

[47] K. Wu, D. Dreef, B. Sun, Y. Xiao, Secure data aggregation without persistent cryptographic operations in wireless sensor networks, Ad Hoc Networks 5 (1) (2007) 100—111.

[48] H. Sanli, S. Ozdemir, H. Cam, Srda: secure reference-based data aggregation protocol for wireless sensor networks, in: IEEE VTC Fall Conference, 2004.

[49] Y. Yang, X. Wang, S. Zhu, G. Cao, Sdap: a secure hop-by-hop data aggregation protocol for sensor networks, in: ACM MOBIHOC'06, 2006.

[50] S. Ozdemir, Secure and reliable data aggregation for wireless sensor networks, LNCS 4836, 2007, pp. 102—109.

[51] S. Ozdemir, Functional reputation based reliable data aggregation and transmission for wireless sensor networks, Elsevier Comput. Commun. 31 (17) (2005) 3941—3953.

[52] D. Westhoff, J. Girao, M. Acharya, Concealed data aggregation for reverse multicast traffic in sensor networks: encryption key distribution and routing adaptation, IEEE Trans. Mobile Comput. 5 (2006) 1417—1431.

[53] I. Rodhea, C. Rohner, n-lda: n-layers data aggregation in sensor networks, in: 28th International Conference on Distributed Computing Systems Workshops, 2008.

[54] W. Zhang, Y. Liu, S. Das, P. De, Secure data aggregation in wireless sensor networks: a watermark based authentication supportive approach, Elsevier Pervasive Mobile Comput. 4 (2008) 658—680.

[55] J. Domingo-Ferrer, A provably secure additive and multiplicative privacy homomorphism, in: Information Security Conference, 2002.

[56] B. Sun, N. Chand, K. Wu, Y. Xiao, Change-point monitoring for secure in-network aggregation in wireless sensor networks, in: IEEE Global Telecommunications Conference (GLOBECOM 2007), 2007.

[57] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks, IETF Request For Comment (RFC 4944), Sep. 2007.

[58] N. Kushalnagar, G. Montenegro, C. Schumacher, IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, IETF Request For Comment (RFC 4919), Aug. 2007.

[59] T. Watteyne, A. Molinaro, M. G. Richichi, M. Dohler, From MANET to IETF ROLL standardization: a paradigm shift in WSN routing protocols, IEEE Communications Surveys and Tutorials, to appear.