Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Ad Hoc Networks xxx (2006) xxx–xxx
Ad Hoc Networks
www.elsevier.com/locate/adhoc

TinyPEDS: Tiny persistent encrypted data storage in asynchronous wireless sensor networks

Joao Girao [a], Dirk Westhoff [a,*], Einar Mykletun [b], Toshinori Araki [c]

[a] *NEC Europe Ltd., 69115 Heidelberg, Germany*
[b] *University of California, Irvine, United States*
[c] *NEC Internet Systems Research Labs, Kawasaki, Japan*

Received 5 May 2006; accepted 25 May 2006

Abstract

In wireless sensor networks there is a need to securely store monitored data in a distributed way whenever it is either not desired or simply not possible to transmit regional volatile information to an authorised recipient in real-time. In particular, for wireless sensor network applications with an asynchronous character, the wireless sensor network itself needs to store the monitored data. Since nodes may disappear over time, a replicated and read-protected, but yet space- and energy-efficient, data storage is mandatory. In this work we provide and analyse an approach for a *tiny Persistent Encrypted Data Storage* (*tinyPEDS*) of the environmental fingerprint for asynchronous wireless sensor networks. Even if parts of the network are exhausted, restoring rules ensure that, with a high probability, environmental information from past is still available.

Keywords: Wireless sensor networks; Data encryption; Data storage; Energy consumption; Homomorphic encryption transformation

1. Introduction

When categorizing wireless sensor networks (WSN)s with respect to the frequency and real-time responsiveness with which the environmental information is provided to authorised parties, in principle we face two cases: The first type of WSNs, which we term *synchronous* WSNs are WSNs where the monitored data is fluctual and the system is most likely to be used for some real-time control monitoring. In contrast, we define an *asynchronous* WSN as a network that provides information to an authorised reader only seldomly. Instead, the network continuously monitors and stores environmental data as a function over the time and over the monitored region. Consequently, after a period of monitoring and storing, the WSN contains a fine granular environmental fingerprint. Thus, complementary to the functionalities of a synchronous WSN, an asynchronous WSN also needs to provide components of a distributed database. However, there are several reasons why the implementation of a distributed database for an asynchronous

---

[*] Corresponding author.
E-mail addresses: joao.girao@netlab.nec.de (J. Girao), dirk.westhoff@netlab.nec.de (D. Westhoff), mykletun@ics.uci.edu (E. Mykletun), t-araki@ek.jp.nec.com (T. Araki).

1570-8705/$ - see front matter © 2006 Published by Elsevier B.V.
doi:10.1016/j.adhoc.2006.05.004

WSN is much more challenging than that of its centralised counterpart located at a powerful server. First and foremost, we cannot assume the devices of a WSN to be tamper-resistant. Consequently, assuming that the WSN is deployed in public, untrusted, or even hostile environments, the entries in the database must be protected and concealed. Secondly, the overall storage space of a WSN is extremely limited and to orders of magnitude more restricted compared to its conventional database counterpart. Even a large scaled WSN where thousands or even millions of sensor nodes may serve as storing cells, will not even compare to the storage capacity of a traditional database. Thirdly, since in WSNs energy consumption is the major metric, database entries and database queries need to be translated in terms of number of hops, transmission distance and CPU cycles to accurately estimate the resulting energy consumption of the database operations. Finally, single nodes, or even whole regions of the WSN, may exhaust earlier than others. Consequently, a strategy for replicating the data storing at different devices is advantageous. However, replicated storage of data at multiple sensor nodes also increases the energy consumption for storage, data queries and data response operations. Also, persistent memory, which is an extremely scarce resource in WSNs, should not be wasted with too much redundant information.

It is the objective of this work to provide an in-network approach to securely and reliably store the monitored information of an asynchronous WSN in an aggregated and replicated way, while at the same time keeping the storage process and the query process energy-efficient. Compared to the continuous monitoring and storing activities of the WSN, we assume database queries and database responses to be rather seldom. In case of a disaster, where considerable parts of the network abruptly disappear, it should still be possible to request the content which was originally monitored in the exhausted part of the network. One can even imagine a scenario where only once after the final collection of the sensor nodes, the nodes provide their monitored data to the reader. Under such circumstances our approach provides a reasonable degree of information accuracy and reliability with respect to range queries. Range queries are the queries where only events within a certain range are desired. More precisely, it is the contribution of this work to provide a read-protected collaborative storage management for multi-resolution storage based on the age and/or the region of the monitored data. We provide read-protection during data distribution, data storage and query response. Range queries can be addressed with different granularity according to region and age. Furthermore, responses of the WSN to range queries from an authorised reader can either offer the average over a region and/or the age, or they can offer a specific peak value over the defined region and/or age of the monitored data. For data protection, we apply a specific class of encryption transformations which we show to have value in the context of WSNs.

The rest of this paper is organised as follows. In Section 2 we present related work. In Sections 3 and 4 we present the assumed network model and the addressed threat model and provide an instantiation of an optimal storage policy under such settings. In Section 5 we introduce two classes of homomorphic encryption transformations and in Section 6 we discuss how to apply such encryption schemes in the context of WSNs. Section 7 presents the *tiny Persistent Encrypted Data Storage* (*tinyPEDS*), our approach for a distributed and persistent encrypted data storage for asynchronous wireless sensor networks. Section 8 introduces controlled flooding strategies for database queries and database responses for *tinyPEDS* whereas in Section 9 we provide restoring rules for the disaster case. In Sections 10 and 11 we analyse the security and validate the performance of our encrypted storage architecture. In the Section 12 we outline how to address encrypted comparison. Section 13 concludes the work and gives an outlook.

2. Related work

Madden et al. in [15] and Hellerstein et al. in [13] provide an SQL-based query model for in-network aggregation in WSNs addressing specific monitoring durations of the network. Queries address monitoring periods in the present and in the future. However, although in [13] the concept of storage points allows a buffered streaming view of recent data, the fully-fledged architecture to store monitored data of an event in the past within the WSN is not addressed in their work.

The problem of how to use the limited persistent storage capacity of a sensor node to store sampled data effectively has been discussed by Tilak, Abu-Ghazaleh and Heinzelmann in [21]. The authors provide a cluster-based collaborative storage approach and compare it to a local buffering technique. Collaborative storage is a promising approach for storage management because it enables the use of spatial data aggregation and redundancy control among neighboring sensors to compress the stored data and to optimize the storage use. However, the concealment of the data stored is not in the focus of their work.

In [10, 24, 3] some of the authors of this paper provided security concepts for a *synchronous* WSN. Both approaches provide end-to-end encryption for real-time responsive reverse multicast traffic from the monitoring nodes to the sink node. The essential difference between approaches in [10] respectively [24 and 3] is that they provide end-to-end encryption with respect to different types of in-network processing for the aggregating and forwarding intermediate nodes. In [10, 24] we provided encryption in presence of in-network processing based on additive operations, whereas in [3] we provided a concealed transmission of real-time data if comparison operations are performed at the aggregating intermediate nodes. For the first, the applied cryptographic technique to ensure end-to-end encryption is an *additive privacy homomorphism* like e.g. proposed by Domingo-Ferrer in [9] whereas the latter is based on the *order preserving encryption scheme* (*OPES*) by Agrawal et al. [2].

Recently Castelluccia, Mykletun and Tsudik presented an efficient approach for the aggregation of encrypted data in wireless sensor networks [4] which is also based on the additively homomorphic property of an encryption scheme. This approach is proved to be perfectly secure. It uses different keys per sensor at the cost of a mandatory transmission of the sensors' ID list of the involved monitoring nodes. Although, when applied to large scaled WSNs with a flat structure, such a requirement contradicts with the aim of reducing the message size and easily results in unacceptably large messages, we believe that for cluster-based WSNs the approach is suitable.

However, all the aforementioned approaches [4,10,3] lack in the sense that due to their storage requirements of sensitive key material they provide only a moderate level of system security. Consequently, in [17] we also considered asymmetric approaches where only the public key needs to be stored on the non-tamper resistant devices.

The above contributions provide valuable building blocks to come up with an encrypted data storage for asynchronous and real-time uncritical WSNs. However, the fully fledged distributed and encrypted long term data storage architecture is still far from being realised.

3. Network and threat model

The WSN considered in this work is static and densely distributed. It is presented by a graph $\mathscr{G} = (\mathscr{N}, \mathscr{L})$ with $|\mathscr{N}|$ nodes and $|\mathscr{L}|$ links. Each node represents a wireless sensor node, e.g., a Mica-z mote, and each link represents a bidirectional communication channel over a shared medium, e.g., the RF channel specified by IEEE 802.15.4 (security suite Null) [5,22]. There is one single stated node R, the reader device, with virtually unlimited power and storage capacity which may be mobile. After a period of monitoring and storing within the WSN the reader device is placed randomly but preferably in the center of the plane covered by $\mathscr{G}$. Per epoch τ during the lifetime of the WSN, a set of aggregator nodes $\mathscr{A}_\tau$, a set of forwarding nodes $\mathscr{F}_\tau$, a set of sensing nodes $\mathscr{S}_\tau$ and a set of idle or sleeping nodes $\mathscr{I}_\tau$ with $\mathscr{A}_\tau \cap \mathscr{F}_\tau \cap \mathscr{S}_\tau \cap \mathscr{I}_\tau = \emptyset$ and $\mathscr{A}_\tau \cup \mathscr{F}_\tau \cup \mathscr{S}_\tau \cup \mathscr{I}_\tau = \mathscr{N}$ is elected by the network. Let x_{in} and x_{out} be the number of incoming and outgoing messages at a node x's network layer. If $(x_{\text{in}}, x_{\text{out}}) = (0, 1)$, then $x \in \mathscr{S}_\tau$, if $(x_{\text{in}}, x_{\text{out}}) = (1, 1)$, then $x \in \mathscr{F}_\tau$, and finally if $(x_{\text{in}}, x_{\text{out}}) = (n, 1)$, $n > 1$ then $x \in \mathscr{A}_\tau$. The latter nodes compute the aggregation function $\text{out} = f(\text{in}_1, \ldots, \text{in}_n)$ on the received data in_i, $i = 1, \ldots, n$ with $f{:}\{0,1\}^k \times \ldots \times \{0,1\}^k \to \{0,1\}^{k+l}$ and $k + l \ll n \cdot k$. For R, $(x_{\text{in}}, x_{\text{out}}) = (n, 0)$. All $x \in \mathscr{I}_\tau$ are not available for sensing and routing in epoch τ. At epoch $\tau + 1$, $\mathscr{A}_{\tau+1}$ varies from $\mathscr{A}_\tau$, where $|D|$ with $D = \mathscr{A}_\tau \cap \mathscr{A}_{\tau+1}$ is a metric for the quality of the aggregator nodes' election process. The smaller the $|D|$, the more energy balanced the aggregator process tends to be. Consequently, also $\mathscr{F}_{\tau+1}, \mathscr{S}_{\tau+1}$ and $\mathscr{I}_{\tau+1}$ may differ from their counterparts in epoch τ. Ideally, $\mathscr{A}_\tau \cup \mathscr{F}_\tau$ form a minimum dominating set connecting each node $\overline{\mathscr{N} \cap \mathscr{I}_\tau}$ such that there exists at least one bidirectional path between any pair of nodes. We term $\mathscr{A}_\tau \cup \mathscr{F}_\tau$ the connected backbone of the WSN in epoch τ. Since finding the minimum dominating set is an NP complete problem, heuristics are needed here.

The attacker model we assume results from the specific device restrictions, as well as from the properties of the shared medium. An attacker can eavesdrop traffic on the wireless channel, it can read data from the sensor nodes memory (since we do not assume nodes to be equipped with tamper-resistant units) and finally, it can monitor environmental data. We are considering large scaled and relatively permanent WSNs. However, we assume that the attacker's capability to monitor environmental data is restricted due to the monitoring range and/or due to the duration of monitoring. From the above observations we follow that the classical Dolev–Yao threat model [8] does not hold in the world of WSNs. It is essential to broaden this model to also address the capturing of sensitive data which is stored in the communicating end-points.

4. Disaster model and collaborative storage policy

Another aspect to consider is adapted to handle secure storage in a disaster-prone environment. Clusters of nodes might die due to what we classify as "disastrous" events. This differs from WSNs in which nodes die uniformally, typically due to energy exhaustion or malfunction. We do not exclude this type of disappearing from our thinking but the goal of a storage strategy in such a setting should be to replicate the data in such a manner that it is likely to survive regional limited disasters, while minimizing the energy costs associated with the replication of data.

Disasters are expected to occur only rarely, if at all. On the contrary, the typical querying of the environment should be supported during regular behavior. At the time a disaster occurs, we assume that the WSN administrator is notified about it and immediately tries to retrieve as much information from the network as possible. We envision large scale WSNs in which a disaster might take out a significant portion of the deployed sensors. A second assumption we make is that we have an estimate of the maximum damage range of possible disasters. By using this knowledge, we can determine the distance needed between two nodes, one replicating the other's data, by which at least one of the nodes will survive a disaster. Let r represent the disaster radius, then a pair of nodes need to be stored at a distance of at least $\gamma = 2r$ apart. We now describe a replicated storage policy optimized for restoring and energy saving under such settings. There is a cost associated with every additional node at which a sensor's data is replicated. An optimal replication strategy should therefore ensure that data is only replicated as much as is necessary to retain it after a disaster. Obviously, the minimum number of replication nodes is one, and this node should be located at least γ distance away from the originating sensor and not much further to save transmission energy. Any larger number of replication nodes would imply a larger cost, although it would most likely increase the robustness of the WSN. To summarize: Under the assumptions

- *Assumption 1*: due to possible disasters, nodes tend to die in clusters, and
- *Assumption 2*: before the WSN is rolled-out we have some way of approximating the upper radius r of a disaster area, within which nodes are disabled.

The most optimal (energy) strategy is to pair up every node $x \in \mathcal{N}$ with a corresponding replicating node $y \in \mathcal{N}$, such that $dist(x,y) > \gamma$, thus ensuring that the data originating at x survives a disaster.

Notice that the query model for distributed database entries of the WSN is different prior and after a disaster strikes. In the former, we envision a query as selecting the aggregate from a specific age and region to learn its value, while in the latter, the query might ask for any information that still resides in the network, such as to salvage as much as possible. We are very well aware of the fact that the energy-efficiency of a distributed database query, beside the concrete storage policy, also depends on the parameters which describe the concrete WSN topology. Parameters like cluster size, number of clusters, cluster levels in the hierarchy, as well as the nodes' transmission range may significantly vary with respect to the concrete WSN application. They impact the effectiveness of the database query process. However, although we continue to describe our approach for a specific *tinyPEDS* friendly set of WSN parameters, we hereby point out that *tinyPEDS* is adaptable to nearly all kinds of WSN topologies.

5. Encryption schemes

Next we introduce two classes of encryption transformations that we apply in the remainder of this work to the *tinyPEDS* approach. Their application ensures the encryption of environmental data for a concealed and indeed space-saving data storage. We assume two *additively* privacy homomorphisms, say PH_s and PH_a, which we define as follows:

Definition 1 (*symmetric additive privacy homomorphism* (PH_s)). Let an encryption transformation be

$E{:}K \times Q \rightarrow R$ and the corresponding decryption function be $D{:}K \times R \rightarrow Q$. Given $a_1, a_2 \in Q$ and $k \in K$, a PH_s shall be based on a symmetric key k. It provides

$$a_1 + a_2 = D_k(E_k(a_1) \oplus E_k(a_2)). \tag{1}$$

In Definition 1 the symbol "+" represents an additively operation on words from the plaintext alphabet and the symbol "$\oplus$" represents the corresponding additive operation on words from the ciphertext alphabet.

Definition 2 (*asymmetric additive privacy homomorphism* (PH_a)). Let an encryption transformation be $E{:}K_p \times Q \rightarrow R$ and the corresponding decryption function be $D{:}K_q \times R \rightarrow Q$. Given $a_1, a_2 \in Q$ and $(p, q) \in (K_p, K_q)$, a PH_a is based on a public/private key-pair (p, q). It provides

$$a_1 + a_2 = D_q(E_p(a_1) \diamondsuit E_p(a_2)). \tag{2}$$

In similarity with Eq. (1) from the Definition 1 for the PH_a, the symbol "$\diamondsuit$" in the Definition 2 is the additive operation on words from the ciphertext alphabet.

Due to its homomorphic properties, a direct application of such schemes for simple encryption purposes is only of moderate interest since malleability may destroy chosen-ciphertext security. However, for some security concepts exactly this homomorphic feature is a necessary prerequisite. In the remainder of this work we apply the PH_s proposed by Castelluccia, Mykletun and Tsudik [4] with

$$E_k(a) = a + k \bmod m \tag{3}$$

and

$$D_k(a) = E_k(a) - k \bmod m \tag{4}$$

where $a \in [0, m-1]$ (respectively Q) and $k \in [0, m-1]$ (respectively K). Let $c_1 = E_{k_1}(a_1)$ and $c_2 = E_{k_2}(a_2)$ with $k = k_1 + k_2$ and $a_1, a_2 \in [0, m-1]$, then $D_k(c_1 + c_2) = a_1 + a_2$. Note that k is a keystream that can be generated by a streamcipher, such as RC4, keyed with a node's secret key and a unique nonce.

One candidate for a PH_a which fulfills the requirement of Eq. (2) is the encryption transformation proposed by Okamota and Uchiyama [18]. Their scheme provides a strong security level, namely a security as secure as factoring. However, the minimum size of each plaintext a with $|a| \leqslant 341$ bits is always $|E_k(a)| = 1024$ bits and the execution times for encryption are approximately two times that of an ECC signature generation with a key size of 163 bits which is in the range of 6s to 10s on the Mica-z Motes reference platform. Compared to the process of transmitting or receiving over a moderate transmission distance, the Okamoto and Uchiyama encryption consumes roughly ten times more energy based on the same amount of data. The addition operation on encrypted data is neglectable with respect to its energy-consumption. Other candidates for a PH_a are presented in [19], namely an embodiment of the Naccache and Stern cryptosystem, an elliptic curve version of the Okamoto and Uchiyama cryptosystem and an encryption transformation by Paillier. A more promising PH_a candidate for the requirements of an energy-restricted WSN is the appliance of the ElGamal public-key encryption [16] on elliptic curve (E) points [17]. The EC-ElGamal encryption scheme is based on the EC discrete logarithm problem (ECDLP). Such a choice reduces the size of the ciphertext to two times the key-length which is typically 163 bits when using standard elliptic curves. We apply

$$M = map(a) \tag{5}$$

$$E_p(M) = (R, S) \quad \text{where } R = kG, \quad S = M + kY \tag{6}$$

with the public key (E, p, G, Y) (respectively K_p) with $G, Y \in F_p$ and G be a generator point. The function $map()$ is a deterministic mapping function used to map plaintext values a into "plaintext" curve points M and vice versa such that $map(a_1 + a_2) = map(a_1) + map(a_2)$. Decryption subsequently applies the reverse mapping function $rmap()$

$$D_q(E_p(M)) = -xR + S = -xkG + M + xkG \tag{7}$$

$$a = rmap(M) \tag{8}$$

with the private key (respectively K_p). In [17] we presented candidates for a mapping function sustaining the homomorphic properties of the encryption scheme. For an example of a mapping function we refer to Appendix A.1.

6. Discussion on the usage of PHs

The additive homomorphic property of a PH is a feature which is of particular relevance for an appliance in WSNs. Properly used, it provides the conceptual framework to efficiently conceal the environmental fingerprint of the covered region as a function over the time by at the same moment considering the extreme restrictions of the destination platform and on the radio. Obviously, in such a restricted environment there is the ultimate need to collaboratively store and to collaboratively transmit data in a condensed and aggregated manner, while still providing an appropriate degree of information to the authorised reader. In this context please note that e.g. for the RFM TR 1000 radio transreceiver and an Atmel ATmega 128L microcontroller the relationship between energy consumption for computation and communication is 1 μJ to transmit a single bit, 0.5 μJ to receive a single bit, and 8 nJ for processing an instruction [14]. This results in a ratio of approximately 190 for communication to computation which documents that communication is considerably more expensive due to energy consumption than computation.

Considering and comparing the usage of a PH_a and a PH_s in a WSN, a PH_a frequently not only provides a better security with respect to the hardness of the underlying mathematical problems, it also offers a better system security due to the storage policy of sensitive data. Here, even when considering a WSN with non tamper-resistant devices, the storage of the public key p on the nodes does not reveal any sensitive information in case a set of nodes gets corrupted. In contrast, when applying PH_s to WSNs the symmetric keys k are stored at the (most-likely) non-tamper resistant devices. However, for the latter scheme, the size of a ciphertext resulting from one byte plaintext depends on the chosen modulus m and is only of three bytes to four bytes [4] compared to a 41 bytes ciphertext when using e.g. the EC-ELGamal reference PH_a with a 163 bit key length. Consequently, since the message size linearly impacts the energy consumption for transmission, a PH_s is preferable from this viewpoint. With seven bytes signaling data, a PH_s encrypted TinyOS packet respectively an IEEE 802.15.4 packet are of size 10–11 bytes. However, a PH_a encrypted message of a total size of 48 bytes also fits into a single TinyOS respectively IEEE 802.15.4 packet. Nevertheless, due to its length, a PH_a encrypted message causes comparably more energy consumption for transmission than a PH_s encrypted one. Finally, even though compared to other PH_a candidates the encryption, addition and decryption for the EC-ElGamal are much more economic in terms of energy, the EC-ElGamal should be applied as less as possible.

Another observation that impacts the design decisions for an encrypted and reliable collaborative data storage system for WSNs is the tendency that, with a broader coverage and a condensed storage of the monitored data due to range or age, there might be a stronger need to protect this information with a higher degree of security. On the contrary, an attacker who is physically located within the WSN, and who can either eavesdrop the traffic from neighboring nodes or sense environmental data within a small region itself, gains only a very limited insight of the whole system knowledge. Furthermore, if the attacker is located within the WSN only for a minor fraction of the WSN's whole lifetime her gain is even more limited.

7. Encrypted and aggregated data storage

Under the aforementioned requirements, limitations and observations, we propose an approach for an encrypted and aggregated collaborative data storage in WSNs which provides some striking advantages:

1. stored data is encrypted and even the storing node cannot decrypt the ciphered values,
2. transmission costs for a collaborative and distributed data storage are minimised,
3. persistent storage space is balanced over multiple sensor nodes and reduced close to the minimum,
4. even if parts of the network are exhausted, the remaining nodes of the WSN, with a high probability, still contain information to restore the database entries which got lost,
5. nodes know from which region and for which epoch they store data. However, they do not know the values they are storing,
6. since a node is aware from which region it stores data, it also knows if a distributed data query from the reader device is addressed to it or not.

The principle idea of the *tiny Persistent Encrypted Data Storage* (*tinyPEDS*) approach is to apply the PH_s and the PH_a to aggregate encrypted data and to ensure due to their nested arrangement a higher system security for condensed information. We apply as PH_s the encryption scheme from Castelluccia et al. and as PH_a the EC-ElGamal encryption scheme.

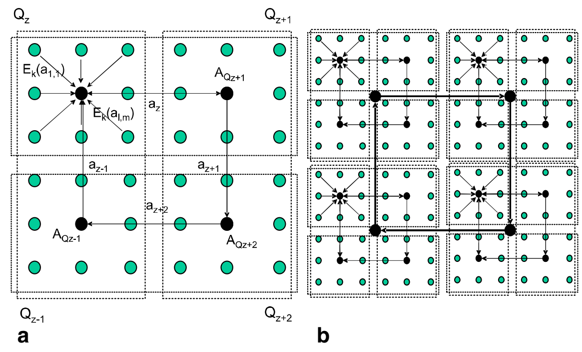


Fig. 1. (a) *TinyPEDS* cluster structure, and (b) hierarchical usage of *TinyPEDS* with quarters and subquarters.

After the WSN's roll-out, initially subdivide the WSN into clusters[1] $\mathcal{Q}_z$, $1 \leqslant z \leqslant \omega$ with $\mathcal{N} = \bigcup_{z \leqslant \omega} \mathcal{Q}_z$. Fig. 1(a) illustrates this for $\omega = 4$.[2] Elect for each quarter per epoch τ a new aggregator node $A_{\mathcal{Q}_z}$ e.g. by applying a *low energy adaptive clustering hierarchy* (LEACH) [12] derivate or a cluster head election algorithm like proposed in [21]. Since aggregator nodes are physically equal to other nodes due to their heavy work load they have to be re-elected from time to time. If necessary, due to the size of the WSN, iteratively subdivide quarters into subquarters such that ideally it holds

$$\forall A_{\mathcal{Q}_z} : \gamma \leqslant dist(A_{\mathcal{Q}_z}, A_{\mathcal{Q}_{z\pm1}}) \leqslant \gamma + \Delta. \tag{9}$$

In this formula we denote the margin of nodes with a still acceptable distance for replicating data of an adjacent cluster with Δ. An exemplary setting with two levels of a hierarchy is illustrated in Fig. 1(b). We assume an aggregator node $A_{\mathcal{Q}_z} \in \mathcal{A}_\tau$ to be responsible in epoch τ from the moment t to the moment $t+1$ for aggregating the monitored data from time slots θ with $\theta \in [t, t+1]$ received from all sensor nodes $s \in \mathcal{S}_t \cap \mathcal{Q}_z$. Each sensor node $s_{i,j} \in \mathcal{S}_t \cap \mathcal{Q}_z$ with $1 \leqslant i \leqslant l$, $1 \leqslant j \leqslant m$ of an $l \times m$ "dimensioned" quarter $\mathcal{Q}_z$ monitors environmental data $a_{i,j}$ and encrypts, per time slot θ, one characteristic value by applying the PH_s from Castelluccia et al. such that $E_{k_{i,j}}(a_{i,j})$. Per time slot θ, all the sensor nodes transmit their ciphers: $\forall s_{i,j}$:$1 \leqslant i \leqslant l$,$1 \leqslant j \leqslant m$:

$$s_{i,j} \rightarrow A_{Q_z} : E_{k_{i,j}}(a_{i,j}) \tag{10}$$

At the end of each time slot θ of epoch τ the aggregator node $A_{\mathcal{Q}_z}$ of quarter $\mathcal{Q}_z$ computes

$$a_z^\theta = \oplus_{i=1}^{l}(\oplus_{j=1}^{m} E_{k_{i,j}}(a_{i,j})). \tag{11}$$

$A_{\mathcal{Q}_z}$ persistently stores a_z^θ.

At the end of each epoch τ, the aggregator node $A_{\mathcal{Q}_z}$ in addition to Eq. (11) sends the encrypted environmental fingerprint of $\mathcal{Q}_z$ to the aggregator node of quarter $\mathcal{Q}_{z+1}$

$$A_{\mathcal{Q}_z} \rightarrow A_{\mathcal{Q}_{z+1}} : a_z^\theta \tag{12}$$

or, if $z = \omega$, to the aggregator node of the quarter $\mathcal{Q}_1$

$$A_{\mathcal{Q}_z} \rightarrow A_{\mathcal{Q}_1} : a_z^\theta. \tag{13}$$

In an optional setting, a_z^θ can even be aggregated over $[t, t+1]$ from epoch τ before being transmitted to $A_{\mathcal{Q}_{z+1}}$.

When receiving the environmental fingerprint from quarter $\mathcal{Q}_{z-1}$ at the end of epoch τ, the aggregator $A_{\mathcal{Q}_z}$ adds a_z^θ and a_{z-1}^θ and applies the EC-ElGamal PH_a to compute

$$E_p(a_z^\theta \oplus a_{z-1}^\theta). \tag{14}$$

$A_{\mathcal{Q}_z}$ persistently stores this cipher.

The monitoring pattern which was applied implicitely can be described as a cyclic monitoring wave over the quarters from which the collaborative storage pattern is derived. Fig. 1(a) illustrates the traffic flow of *tinyPEDS* for data storage during the WSN's lifetime whereas Fig. 1(b) illustrates the applicability of the *tinyPEDS* approach in a hierarchical setting. More concretely, in a preferable setting of *tinyPEDS* we propose after the θth time slot of epoch τ within the WSN an aggregator node A_{Q_z} to persistently store

$$storage_{[t,t+\theta]} := E_p(a_z^\theta)||E_p(a_z^\theta \oplus a_{z-1}^\theta)||age_{t-1}^{\mathcal{Q}_z \cup \mathcal{Q}_{z-1}}||age_{t-1}^{\mathcal{Q}_z} \tag{15}$$

with

$$age_{t-1}^{\mathcal{Q}_z \cup \mathcal{Q}_{z-1}} := age_{t-2}^{\mathcal{Q}_z \cup \mathcal{Q}_{z-1}} \Diamond E_p(a_z^\theta \oplus a_{z-1}^\theta) \quad \text{for} \quad \theta = t-1 \tag{16}$$

and

$$age_{t-1}^{\mathcal{Q}_z} := age_{t-2}^{\mathcal{Q}_z} \Diamond E_p(a_z^\theta) \quad \text{for} \quad \theta = t-1 \tag{17}$$

whereas

$$age_1^{\mathcal{Q}_z \cup \mathcal{Q}_{z-1}} = E_p(a_z^1 \oplus a_{z-1}^1) \tag{18}$$

---

[1] Note that clusters not necessarily need to be equally shaped nor do they need to be quadratic. However the clusters need to be ordered clockwise and in either case clusters of the same hierarchy level belong to each other.

[2] For a better illustration we are choosing four quarters to be the number of clusters per hierarchy level in the remainder of this work.

and

$$age_1^{\mathcal{Q}_z} = E_p(a_z^1). \tag{19}$$

The symbol "$\|$" denotes the concatenation of two messages. Note that the appliance of the two introduced PHs in such a nested arrangement, like it is proposed for the usage in *tinyPEDS*, holds if and only if the additive operation of the PH_s on ciphertext words fits to the additive operation of the PH_a on plaintext words. More precisely, the symbol "$\oplus$" from the Definition 1 and the symbol "+" from the Definition 2 are the same operations. Also the set Q from PH_s needs to be contained in the set R from PH_a. By using the introduced reference PHs this precondition holds.

8. Query flooding and query response

Next we describe how the reader device requests data from the WSN. More precisely, we discuss how database queries from the reader device can be addressed in presence of the introduced collaborative storage approach and how the *tinyPEDS* WSN architecture handles distributed database queries.

8.1. Controlled query flooding

A distributed database query from the reader has the following type: $query{:=}\langle region, duration, aggregation, TTL, QT\rangle$ where *region* is any subregion of granularity $\mathcal{Q}$ from $\mathcal{N}$, *duration* is any subinterval over the WSN's current lifetime $[t_x, t_y]$ from $[t_0, t_{actual}]$ with $t_0 \leqslant t_x \leqslant t_y \leqslant t_{actual}$ and $aggregation \in \{+,>,<\}$. In Section 12 we show that also aggregation functions of type "<" and ">" can be handled. The parameter *TTL* is the time-to-live field which indicates the flooding range of the query. The query begins with $ttl_{\max}$ representing the maximum hop-distance the query will travel within the WSN. The query type field $QT{:=}\{C,D\}$ allows the sensor nodes to differentiate between a continuous database query ('*C*') and queries which are addressed only in case a disaster appeared ('*D*'). A database query from the device R of type $\langle region, [t_x, t_y], aggregation, ttl_{max}, C\rangle$ is handled by a receiving sensor s as denoted in Algorithm 1. In case of a disaster where major parts of $\mathcal{Q}_z$ were lost, the reader device R sends a query of type $\langle region, duration, aggregation, ttl_{max}, D\rangle$. Such a query is only sent in case a continuous database query did not succeed. We denote the behavior of a sensor node receiving a disaster query in the Algorithm 2.

Algorithm 1. Continuous query

```
if $s \in \mathcal{Q}_z$ AND $\mathcal{Q}_z \subseteq region$ then
    if $ttl_{current} > 1$ then
        $ttl_{current} = ttl_{current} - 1$
        $s \rightarrow *{:}\langle region, [t_x, t_y], aggregation, ttl_{current}, C\rangle$
        if $aggregation = true$ AND $storage_{[t,t+1]} \cap region \neq \emptyset$
        AND $t_x \leqslant t \leqslant t_y$ then
            $s \rightarrow R{:}\langle storage_{[t,t+1]}\rangle$
        end if
    end if
else
    $ttl_{current} = 0$
end if
```

Algorithm 2. Disaster query

```
if $s \in \mathcal{N} \setminus \mathcal{Q}_z$ then
    if $ttl_{current} > 1$ then
        $ttl_{current} = ttl_{current} - 1$
        $s \rightarrow *{:}\langle region, [t_x, t_y], aggregation, ttl_{current}, D\rangle$
        if $aggregation = true$ AND $storage_{[t,t+1]} \cap region \neq \emptyset$
        AND $t_x \leqslant t \leqslant t_y$ then
            $s \rightarrow R{:}\langle storage_{[t,t+1]}\rangle$
        end if
    end if
else
    $ttl_{current} = 0$
end if
```

The fundamental difference between a continuous query type and a disaster query type is that the disaster query, by definition, floods the complementary region of the area where the data was originally monitored. Note that both types of flooding messages, respectively distributed database queries, are also applicable in case the WSN is subdivided in a hierarchy of quarters. Here, an aggregator node of the upper level adopts to the role of a reader at a lower level. Generally a WSN of l levels has 4^l leaf quarters. Typically, *region* can be noted as $region = \langle level_1 \triangleright level_2 \ldots \triangleright level_l \rangle$ with $level_i \in \mathcal{P}(ALL)$ and $ALL := \{\mathcal{Q}_{(i,1)}, \ldots, \mathcal{Q}_{(i,4)}\}$ for $1 \leqslant i \leqslant l$. For example, the database query $\langle \mathcal{Q}_{(1,2)} \triangleright \mathcal{Q}_{(2,1)} \cup \mathcal{Q}_{(2,3)}, [t_x, t_y], aggregation, ttl_{\max}, C\rangle$ results in an energy-saving controlled query flooding like depicted in Fig. 2(a). Its counterpart for the disaster case of nodes which were lost from $\mathcal{Q}_z$ e.g. $\langle \mathcal{Q}_{(1,2)} \triangleright \mathcal{Q}_{(2,1)} \cup \mathcal{Q}_{(2,3)}, [t_x, t_y], aggregation, ttl_{\max}, D\rangle$ results in the controlled flooding as it is illustrated in Fig. 2(b). Although in principle mobile, w.l.o.g. we

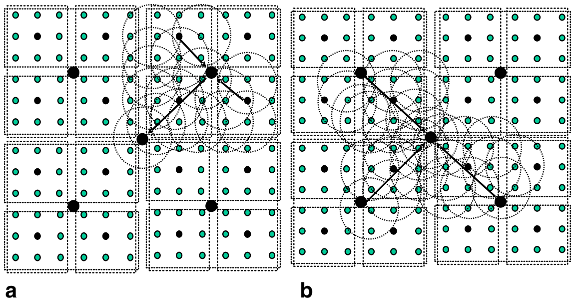


Fig. 2. Continuous query and disaster query.

are assuming that the reader device, when initiating a distributed database query, is located in the center of the WSN. Note that continuous database queries are very energy-efficient, especially for hierarchical WSNs. We will refine this statement in Section 11.

8.2. Aggregated data response

Choosing a PH_a instead of a conventional encryption scheme for encrypting data at the end of each epoch, has a prominant advantage in the phase of the database response. In the most general case, the requested information is distributed over multiple sensor nodes which need to respond to a data-base query. Moreover, these nodes are most likely to be located in each other's neighborhood. Consequently, there is the possibility to perform in-network processing also during the data response phase. With respect to the traffic flow, this phase is similar to our previous work on *concealed data aggregation* for *synchronous* WSNs [10]. The traffic pattern of a distributed database response can be classified as reverse multicast traffic to the reader device. However, since the environmental data within the *tinyPEDS* approach needs to be prepared for long-term storage, in the context of this work we are using a provably secure PH_a instead of a PH_s as it was proposed in previous work for real-time monitoring. Obviously, the transmission of the relatively large cipher of a PH_a encrypted plaintext compared to that of a PH_s encrypted one is a disadvantage. We argue that the number of database queries is comparably marginal to the continuous monitoring and proceeding of secure and reliable distributed data storage. Another argument to defend the choice of a provably secure PH_a with respect to its generally poorer transmission costs is the fact that with an ElGamal encryption over elliptic curve points there is a PH_a candidate available which is characterised by having moderate size of the ciphertext. With a ciphertext size of e.g. 41 bytes, the ciphertext perfectly fits into a single IEEE 802.15.4 standardised packet and thus it is still suitable for a transmission. Obviously, the optimal position of an aggregator node, which is responsible for aggregating traffic of a database response to the reader device, is located between the responding sensor nodes and the reader device. As a heuristic for such a node election, and under the assumption that the reader is located in the center of the WSN, we propose to apply the following rule:

Heuristic: (aggregator election for database response) *Elect* $s \in \mathcal{A}_{\tau_{actual}} \cap level_{l-1}$ *which recently received a database query of type* $\langle$ *region*,$[t_x, t_y]$, *aggregation*, $ttl_{current}, C\rangle$ *with* $region = \langle level_1 \rhd level_2 \ldots \rhd level_l\rangle$ *to aggregate responses to the data query from nodes of* $level_l$.

Sensor nodes which have been elected for a data aggregation of the response traffic according to the above heuristic apply the additive operation $\Diamond$ of the PH_a on the incoming ciphertexts. Each of these ciphers represents a different epoch $t \in [t_x, t_y]$ of the corresponding query. This does not necessarily mean that an aggregator node needs to perform $(t_y - t_x - 1)$ times the operation $\Diamond$ on the received ciphers. Since with an increasing lifetime of the WSN, some sensor nodes will most probably have been responsible for the storage of condensed and ciphered data for more than one epoch, it may happen that they have stored multiple units, say p storage units, e.g. $storage_{[1,2]}$, $storage_{[2,3]}, \ldots, storage_{[p,p+1]}$. W.l.o.g. we use a sequential ordering $t_x \leqslant t_1 \leqslant t_2 \leqslant \ldots \leqslant t_p \leqslant t_y$ here. In such a case a sensor node from $level_l$ aggregates all its query relevant storage units and computes

$$storage_{[1,p]} = \Diamond_{i=1}^{p} storage_{[i,i+1]} \tag{20}$$

before transmitting

$$s \to A_{level_{l-1}} : \langle storage_{[1,p]}, p\rangle. \tag{21}$$

The responsible aggregator node $A_{level_{l-1}}$ is either waiting a pre-defined system time or it continues until the summation of all the received p values is equal to $t_y - t_x$. Subsequently it applies $\Diamond$ to the received ciphers and transmits

$$A_{level_{l-1}} \to R : \langle storage_{[t_x,t_y]}, t_y - t_x\rangle. \tag{22}$$

9. Restoring rules for disappeared quarters

We observe that with a storage policy, such as the one introduced in Section 7, the environmental data which was originally monitored, e.g. in quarter $\mathcal{Q}_{z-1}$, is partly stored in an aggregated and encrypted form together with environmental data from another quarter at the aggregator node $A_{\mathcal{Q}_{z-1}}$, from quarter $\mathcal{Q}_{z-1}$, as well as at the aggregator node $A_{\mathcal{Q}_z}$, from quarter $\mathcal{Q}_z$. This observation can be used for the following restoring rules for any situation where a quarter $\mathcal{Q}_z$ with $1 \leqslant z \leqslant 4$ or a particular $A_{\mathcal{Q}_z}$, which has been elected for epoch t, is exhausted or unavailable.

Restoring Rule 1. A collaborative database query $\langle \mathcal{Q}_z \cup \mathcal{Q}_{z-1}, [t_x, t_y], +, ttl_{\max}, D \rangle$ can also in absence of $\mathcal{Q}_z$ and in particularly in absence of $A_{\mathcal{Q}_z}$ be handled by the remaining WSN as follows: The aggregator nodes from $\mathcal{Q}_z$, $\mathcal{Q}_{z+1}$ and $\mathcal{Q}_{z-1}$, which were responsible in epochs $t \in [t_x, t_y]$, send

$$A_{\mathcal{Q}_{z-1}} \to R : E_p(a^{\theta}_{z-1} \oplus a^{\theta}_{z}) \tag{23}$$

$$A_{\mathcal{Q}_{z+1}} \to R : E_p(a^{\theta}_{z+1} \oplus a^{\theta}_{z+2}) \tag{24}$$

$$A_{\mathcal{Q}_{z+2}} \to R : E_p(a^{\theta}_{z+2} \oplus a^{\theta}_{z-1}). \tag{25}$$

R applies the private key q to the decryption transformation of PH_a to decrypt the incoming ciphers. Subsequently, after applying the decryption function of PH_s, R adds the persistently stored data from the unavailable quarter's direct neighbors $(\mathcal{Q}_{z-1}, \mathcal{Q}_{z+1})$ and subtracts the environmental fingerprint from its opposite quarter $\mathcal{Q}_{z+2}$. Obviously, the final result is equal to the environmental fingerprint from region $\mathcal{Q}_z \cup \mathcal{Q}_{z-1}$ which has been stored in the disappeared quarter $\mathcal{Q}_z$:

$$\begin{aligned} &D_q(a^{\theta}_{z} \oplus a^{\theta}_{z-1}) + D_q(a^{\theta}_{z+1} \oplus a^{\theta}_{z+2}) - D_q(a^{\theta}_{z+2} \oplus a^{\theta}_{z-1}) \\ &\quad = D_q(a^{\theta}_{z} \oplus a^{\theta}_{z+1}) \end{aligned} \tag{26}$$

Restoring Rule 2. A collaborative database query of type $\langle \mathcal{Q}_z, [t_x, t_y], +, ttl_{\max}, D \rangle$ can also in absence of $\mathcal{Q}_z$ and in particularly in absence of $A_{\mathcal{Q}_z}$ be handled by the remaining sensor nodes of the WSN as follows: Apply *Restoring Rule* 1 and subtract from the result $D_k(a_{z-1})$:

$$D_q(a^{\theta}_{z} \oplus a^{\theta}_{z-1}) - D_q(a^{\theta}_{z-1}) = D_q(a^{\theta}_{z}) \tag{27}$$

Both restoring rules hold in cases where only one aggregator node (or quarter) per level and per epoch is exhausted. If two or more aggregator nodes from different quarters of the same level exhaust, data is irrevocably lost.

10. Security analysis

According to the extended Dolev–Yao attacker model, which is in particular applicable to WSNs with non-tamper resistant devices, a complete *tinyPEDS* security analysis includes an evaluation of the applied cryptoschemes, as well as an evaluation of the proposed security architecture. Obviously, the weakest component identifies the security level of the complete system.

Security of the cryptoschemes. The security of the PH_a ElGamal encryption scheme is *based* on the intractability of the discrete logarithm problem in the group G. The group G should be chosen such that (i) G should be relatively easy to apply, and (ii) the discrete logarithm problem in G should be computationally infeasible. The group of points on an elliptic curve over binary fields appears to meet these two criteria. The security of the PH_s proposed by Castellucia, Mykletun and Tsudik is proven to be perfectly secure [4]. The proof is listed in the Appendix A.2.

Security of the architecture. We reduce the security analysis of the *tinyPEDS* architecture to a discussion on the storage of sensitive key material on non tamper-resistant devices. Each sensor node s_i stores the two keys k_i and q. The reader device stores the keys $\sum_{i=1}^{|\mathcal{N}|} k_i$ and the private key p. For the PH_a all sensor nodes s_i need to store the same public key p and only the reader device needs to store the sensitive private key q (We assume the reader device to be equipped with a tamper-resistant module.). Such a key storage reveals no information to an attacker who picks up nodes and reads the public key out of one (or more) randomly chosen sensor node(s). For the PH_s each node s_i stores a different symmetric key k_i it solely shares with the reader device. Consequently, the gain of an attacker when breaking one sensor node is limited to the transmission link between this sensor node and the actual aggregator node of this epoch. All other links between the actual aggregator node and its sensing nodes are not affected.

Even if the attacker picks up a set of sensor nodes his gain is always limited to a small subregion of the WSN. Recall that by applying a PH_s for the continuous monitoring phase instead of using a PH_a, we avoided having to handle disproportional long ciphers to be continuously transmitted over the wireless. Finally, note that the authenticity of the involved sensor nodes is definitively required but out of the scope of this work. Also resilient data aggregation to deal with manipulated environmental data is an issue which is not further considered here. For both we refer to available work on sensor networks from the literature.

11. Performance and simulation

11.1. Performance of the reference PHs

We measured the costs of our reference candidates for PH_s and PH_a, namely the Castelluccia et al. approach and the ElGamal cryptoscheme on elliptic curve points. We analysed their performance in terms of computation for encryption, addition and decryption as well as the resulting size of the ciphertexts. These values are summarised in Table 1. Here $xm + ya$ means x multiplications modulo 32 plus y additions modulo *32*. We assume the plaintext of a PH_s encryption to be 16-bit (which provides a reasonable accuracy for almost all types of monitored data) and for the PH_a encryption to be 32-bit. Such plaintext values reflect the required nested arrangement of the two homomorphic schemes where the plaintext input of the PH_a tends to be larger than the plaintext input of the PH_s. More concretely we are assuming the following parameters:

- **Castelluccia et al.** [4]: 16-bit plaintext, 32-bit modulus, 32-bit ciphertext;
- **EC-ElGamal** [17]: 32-bit plaintext, 163-bit modulus, a mapping function representing monitored values as elliptic curve points and its reverse mapping by conserving the homomorphic feature of the EC-ElGamal scheme.

For the mapping function we assume that each EC-ElGamal encryption at the first-level aggregators corresponds to 10 ciphertexts. Then the decrypted aggregated value will be $32 + 4 = 36$ bits long (4 from log(10)). If we use the $O(\sqrt{z})$ method to balance storage versus computation for the baby-step giant-step brute force reverse mapping function, then we will store 2^{18} pre-computed values at the decryptor, and we will perform on average 2^{17} ECC additions when reverse mapping the plaintext. For the reverse mapping and the EC-ElGamal encryption it takes approximately 3.3 seconds assuming approximately 40,000 point additions per second.

The computation costs and in particular the encryption costs for our reference PH_a illustrate our design decision to use a PH_a encryption only once per epoch contrary to a slot-wise application of the PH_s encryption operation. A concrete ratio of the number of slots per epoch and the pause times between the slots is an application specific configuration which needs to be carefully balanced with respect to the required system security, its monitoring accuracy, as well as the aimed energy-efficiency. We want to point out that in an *asynchronous* WSN with the objective of an encrypted long term data storage execution time is not the major metric. Nevertheless execution time relates to microcontroller instructions which directly translates into energy consumption. However, as we will see below, in particular at the aggregator node, compared to energy consumption for communication this is still marginal. In [11] Gura et al. show how to speed-optimize ECC to be reasonable also in WSNs. They implemented the ECC point multiplication in assembly code using optimized multiplication algorithms, well suited for the underlying hardware. They measure the time to compute xP over a 160-bit ECC curve on an Atmel ATmega128 at 8 mhz to take 0.81 sec. For a 160-bit number, we expect

Table 1
Performance of Reference PHs: PH_s from Castelluccia et al. and PH_a EC-ElGamal

	Encryption	Add	Decryption	Bandwidth [byte]
PH_S (Castelluccia et al.)	1 m + keystream	1 m	1 m + keystream	4
PH_a (EC ElGamal)		$10\|p\|$ (1)		41
	$2\frac{15}{2}\|p\| + \frac{15}{2}\|m\|$		$\frac{15}{2}\|p\| + 5 + map$	

All values are measured in terms of 32-bit moduli computations, i.e. 2m and 3a represent 2 modular multiplications and 3 modular additions, respectively.

240 (3/2 ∗ 160) point additions/doublings. Although we suggest the use of a 163-bit curve this should not cause a large difference. With respect to the *TinyPEDS* EC-ElGamal computation the above point multiplication occurs three times when sensors encrypt, namely to compute $M = mG$ (mapping), $R = kG$ and $S = M + kY$. However, none of the three require a "full" point multiplication, meaning that the constant k is not 163 bits, and for R and S, it is only 80 bits. Theoretically, this would mean that the computation of R and S take approximately 0.41 seconds each, so the encryption could be done in below 1 second.

11.2. Performance of tinyPEDS

When analysing the effect of the *tinyPEDS* architecture with respect to the energy-consumption we have to mainly consider the impact on the physical layer and on the MAC layer, e.g. the IEEE 802.15.4 WPAN MAC protocol. The IEEE 802.15.4 protocol is inherently asymmetric for a PAN coordinator and its neighbor nodes. This also reflects the energy consumption of an aggregator node $A_{\mathcal{Q}_z}$ and the remaining nodes of a *tinyPEDS* quarter. Basically the ratio of an active period to an inactive period is dramatically different for the $A_{\mathcal{Q}_z}$ compared to the sensing and sending nodes s. Whereas in principle an s only needs to be active in transmit mode in one of the *Guaranteed Time Slots* (GTS), the $A_{\mathcal{Q}_z}$ needs to be active in receive mode the whole active period of an IEEE 802.15.4 superframe[3]. This translates into roughly 8 times more energy consumption due to communication for $A_{\mathcal{Q}_z}$ than for a sensing sensor node. Finally, note that the IEEE 802.15.4 data field has variable size and can go up to 102 bytes. This means that both, a PH_s encrypted two bytes plaintext and a PH_a aggregated and encrypted cipher fit into an IEEE 802.15.4 data field and can be transmitted in a single GTS. This indicates that, if properly adapted, the *tinyPEDS* impact on the WSN's overall energy-consumption is indeed moderate. Both, data traffic during continuous monitoring and data traffic for aggregated data response do perfectly fit into the MAC layer architecture. Recall that distributed database queries are expected to occur only seldomly. Thus, a PH_a encrypted plaintext needs to be transmitted only rarely.

To conclude, for an indeed energy-efficient processing of the *tinyPEDS* architecture, one needs to properly adapt the *tinyPEDS* settings to the configuration possibilities of the MAC layer protocol, e.g. the IEEE 802.15.4 standard. In particular this means:

- The number of nodes within one cluster $\mathcal{Q}_z$ should correspond to the number of GTS within an IEEE 802.15.4 superframe.
- A *tinyPEDS* time slot θ should correspond to an IEEE 802.15.4 superframe.
- To avoid collisions over the wireless one should reduce communication during the *Contention Access Period* (CAP) slots as far as possible. In particular this means that the epochs τ for which a node is elected as aggregator node should be relatively large to reduce the ratio of collision endangered CAP communication for the re-synchronisation of the cluster members after an election.
- According to the concrete WSN application's requirements due to the monitoring accuracy, the inactive period of an $A_{\mathcal{Q}_z}$ should be as long as possible, or, vice versa, its activity period should be as short as possible.

Recall that the relationship between communication and computation is approximately 190:1. With such a ratio in mind one can easily infer that the above listed cross-layer design decisions are to orders of magnitude more energy-saving compared to the negative energy impact for computations at the microcontroller for the two reference PHs (Fig. 3).

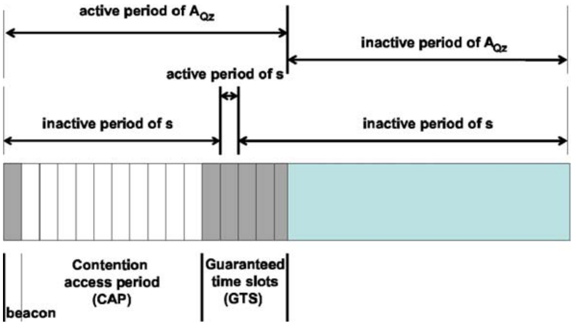


Fig. 3. Superframe structure of IEEE 802.01.4 and energy asymmetry of aggregator node vs. sensing nodes.

---

[3] This is true when $A_{\mathcal{Q}_z}$ is synchronised with its neighbors and there is no need to communicate within the Contention Access Period (CAP).

Finally note that with a setting like proposed in Eq. (15), the *tinyPEDS* architecture translates into the storage size of four times 41 bytes per epoch. Under the assumption that e.g. one half of the Mica-z Mote's 4kB RAM is available for the storage of the environmental fingerprint this means that each sensor node is enabled to be approximately 12 epochs in the role of an aggregator node. We expect the remaining RAM to be occupied for running *tinyPEDS*, PH_s, PH_a and various tinyOS components.

11.3. Simulation

For the simulation, we used the GloMoSim simulator version 2.0 [25]. The simulation setup is listed in Table 2. The simulation results validate the following aspects:

- The storage policy is robust even in presence of a major fraction of exhausted nodes.
- Database queries and database responses work properly in presence of a fraction of exhausted nodes.
- Continuous database queries are efficient in terms of relayed messages.

Figs. 4 and 5 show fractions of involved sensor nodes (marked black) for two particular database queries querying data addressed to the gray marked regions. The considered WSN has three hierarchy levels. The database query from the reader, for Fig. 4, is $\langle \mathcal{Q}_{(1,1)} \triangleright \mathcal{Q}_{(2,4)} \triangleright \mathcal{Q}_{(3,4)}, [t_x, t_y], +, 20, C \rangle$ whereas, for Fig. 5 it is $\langle \mathcal{Q}_{(1,1)} \triangleright \mathcal{Q}_{(2,4)} \Delta \mathcal{Q}_{(3,2)} \cup \mathcal{Q}_{(3,4)}, [t_x, t_y], +, 20, C \rangle$. Since intermediate nodes know their relative positions with respect to the query destination they can control the flooding by dropping query messages in case they belong to quarters which are not relevant for a particular database query. This ensures for the continuous case a moderate propagation of the database query only to the relevant regions and subregions of the WSN as shown in the Figs. 4 and 5.

Table 2
GloMoSim simulation parameters

WSN size	400 × 400
Quadrant size	50
Number of nodes	240–407
Node's transmission range	50
Hierarchy levels	2–3
Number of quarters	4
Node placement	Random
Radio layer	CSMA
Propagation pathloss	Two-way

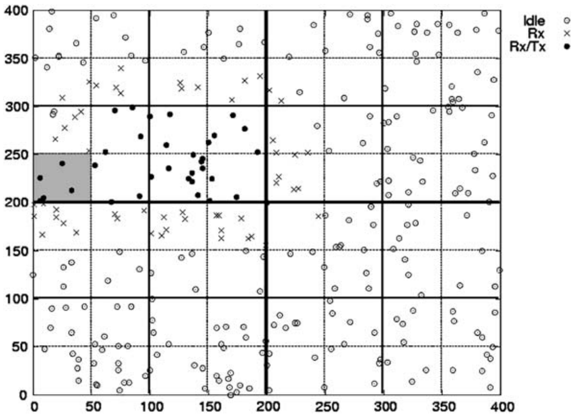


Fig. 4. Controlled flooding of the continuous database query $\langle \mathcal{Q}_{(1,1)} \triangleright \mathcal{Q}_{(2,4)} \triangleright \mathcal{Q}_{(3,4)}, [t_x, t_y], +, 20, C \rangle$ in a WSN with $l = 3$ and $\omega = 4$.

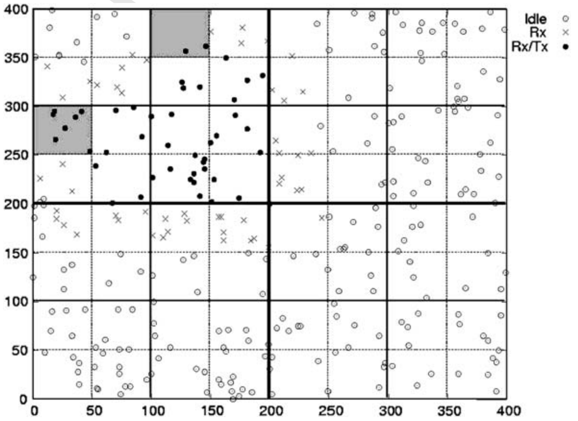


Fig. 5. Controlled flooding of the continuous database query $\langle \mathcal{Q}_{(1,1)} \triangleright \mathcal{Q}_{(2,4)} \triangleright \mathcal{Q}_{(3,2)} \cup \mathcal{Q}_{(3,4)}, [t_x, t_y], +, 20, C \rangle$ in a WSN with $l = 3$ and $\omega = 4$.

Figs. 6 and 7 illustrate for a WSN with three levels of hierarchy how different fractions of exhausted nodes influence the connectivity within the WSN. The simulation start with 10.6 resp. 18.2 neighbouring nodes. Each curve represents the nodes' connectivity within the region size of one specific level. E.g. "Level 3" provides the overall connectivity of nodes within one single quarter of level three. Obviously the curve for "Level 1" illustrates the connectivity between all nodes of the WSN whereas deeper levels illustrate the connectivity within specific quarters of the WSN. The results acknowledge that in smaller

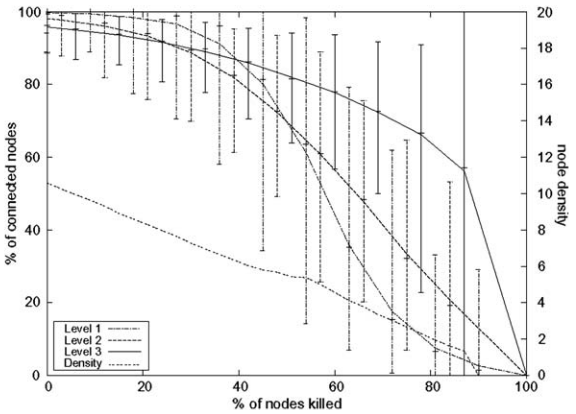


Fig. 6. Connectivity graphs for quarters of three different hierarchy levels with 240 nodes in total. The smallest quadrants are of unit size 50 × 50 and the size of the WSN is 400 × 400. The transmission radius is 50 units resulting in an initial average number of 10.6 neighbors. The density curve shows the average number of one-hop neighbors per node.

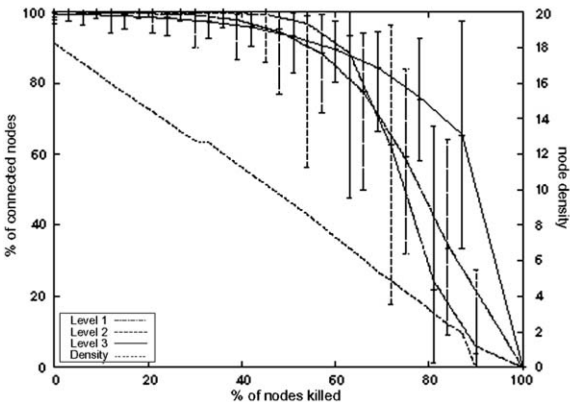


Fig. 7. Connectivity graphs for quarters of three different hierarchy levels with 407 nodes in total. The smallest quadrants are of unit size 50 × 50 and the size of the WSN is 400 × 400. The transmission radius is 50 units resulting in an initial average number of 18.2 neighbors. The density curve shows the average number of one-hop neighbors per node.

regions, with shorter average path length, end-to-end connectivity remains higher than for the whole WSN. Generally we can observe that the storage policy of the WSN needs good connectivity at all levels. On the contrary, for database requests and database responses in its continuous case the connectivity at level one is not as important as the connectivity at the deeper levels.

The illustrated results validate that, under such settings, even up to 40% of the sensor nodes may exhaust over the time to still let *tinyPEDS* proceed well. This holds for the continuous data monitoring and storage process but also for database queries and responses. If we want to translate this critical point into the minimum number of neighbors under such settings, each node should have at least six neighbors alive to allow *tinyPEDS* to work properly (connectivity in the range of 90%). With an increased ratio of exhausted nodes we observe the tendency that database queries and database responses may still have acceptable throughput whereas continuous replicating of the monitored data to a neighboring quarter may be more and more difficult. Consequently, there will be a phase during the lifetime of the WSN where a replicated data storage is already error-proned although the query and response process to request data from the past may still work properly.

12. Encrypted comparison

TinyPEDS, as it is described so far, offers a condensed storage of encrypted data which represent the sum of the monitored environmental values. This has most value when computing the average value over the time or region or when performing movement detection on encrypted data. However, in [20] Rivest et al. have shown that any privacy homomorphism, no matter if it is deterministic or probabilistic, when being homomorphic with respect to the comparison operations, it is insecure against ciphertext only attacks. Consequently one has to find other solutions when supporting database queries based on comparison operations. For this objective we propose to use the *order preserving encryption scheme* (*OPES*) [2] which has been originally proposed for ordering in relational databases and which offers security against ciphertext only attacks. In previous work [3] we have shown that the *OPES* scheme is, in principle, portable on the sensor nodes. Nevertheless, due to the codesize and the memory occupied, running PH_s, PH_a and *OPES* on the same sensor node platform is unrealistic. Instead, we propose to pre-configure the nodes before their roll-out: half of the sensor nodes run the code for PH_s plus PH_a and the remaining half of the sensor nodes runs *OPES*. With the roll-out, both types of nodes are equally distributed over the covered region. They span two overlapping types of WSNs, namely WSN_{PH} and WSN_{OPES}. The WSN_{PH} is responsible for a condensed and encrypted sum representation whereas the WSN_{OPES} does the same for the minimum or maximum. Nevertheless, both WSNs are not fully disjunctive since nodes from both WSN forward traffic also from the other WSN. With such a roll-out, database queries can also address comparison operations by applying exactly the same database queries and database responses as introduced in Section 8.

13. Conclusion

We introduced the concept of persistent encrypted long-term data storage in asynchronous WSNs. We distributed and stored the environmental fingerprint of an area covered by a WSN in a condensed and concealed manner by applying two types of homomorphic encryption transformations. For encryption during aggregation we used a symmetric privacy homomorphic encryption scheme, whereas for a long term replicated storage of the data we applied an asymmetric privacy homomorphism which obviously is the better choice from the security engineering viewpoint. To also handle database queries on minimum or maximum operations we propose to run *OPES* for a fraction of the sensor nodes and roll-out two overlaying types of networks.

14. Uncited references

6,7,23.

Acknowledgements

The authors would like to thank Satoshi Obana and Marc Stoecklin for helpful comments. This work is supported in part by the European Commission within the STREP UbiSecSens (http://www.ist-ubisecsens.org). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the UbiSecSens project or the European Commission.

Appendix A

A.1. Homomorhic mapping function

In order to utilize the additive homomorphic property of the EC-ElGamal encryption scheme, referred to in Section 5, we also require a corresponding homomorphic mapping function. Specifically, we need the following property to hold: for all $a_1,a_2 \in F_p, map(a_1+a_2) = map(a_1) + map(a_2)$. The homomorphic mapping function that we use is similar to the one proposed by VoteHere in [1], and is based upon using multiples of a generator element to represent mapped values. The concept of generators is familiar in finite fields and all its properties apply to elliptic curves, including the fact that a generator point that is continuously added to itself will enumerate all elements in the field. Our approach is to map plaintext value j to the EC point jG, and reverse mapping entails extracting j from jG. This realizes our desire for a homomorphic mapping function as the following operations hold: for $i,j \in F_p,(i+j)G = iG + jG$, where p is the prime defining the curve. The value $i = 0$ is represented by the point at infinity, which is the identity element in elliptic curve groups. As can be seen, the mapping function only involves addition of points, and can be optimized through the use of the add-and-double algorithm (the equivalent of square-and-multiply) and pre-computation of points at regular intervals.

A.2. Security Proof of PH_s

The homomorphic encryption scheme PH_s is very similar to a xor-based stream cipher. Its security can be proven using a similar proof. The security relies in two important features: (1) the key-stream changes from one message to another and (2) all the operations are performed modulo an integer M. These two features protect our scheme from frequency analysis attacks. In fact, it can be proven that our scheme is *perfectly* secure.

Proof. For plaintext space m, key-stream space K, let $K = |m|$, $a \in [0,m-1]$, $c \in [0,m-1]$. Set $k^* = c - a(\text{mod}m)$. Then:

$$\begin{aligned} kK[Enc(k,a,m) = c] &= kK[k + a = c(\text{mod}m)] \\ &= kK[k = c - a(\text{mod}m)] \\ &= kK[k = k^*] \end{aligned}$$

If we assume that the maximum number of ciphertexts to be added is n and that each plaintext is l-bit long, we must have $m = 2^{l+\lceil\log(n)\rceil}$, i.e., $|m| = l + \lceil\log(n)\rceil$. If $c_i = (a_i + k_i)$, then the probability that $c_i \in [0,2^l-1]$ is twice the probability that $c_i \in [2^l,m-1]$. More specifically, we have: $k \quad K[k = k^*] = 1/(2^l + m)$ if $c > 2^l$ and $k\ K[k = k^*] = 2/(2^l + m)$ if $c < 2^l$.

Since these two equations hold for every $a \in m$, it follows that for every $a_1,a_2 \in M$ we have

$$kK[Enc(k,a_1,m) = c] = kK[Enc(k,a_2,m) = c]$$

which establishes perfect security of our scheme. □

References

[1] J.M. Adler, W. Dai, R.L. Green, C.A. Neff, Computational details of the votehere homomorphic election system, in: ASIACRYPT 2000, Kyoto, Japan, December 2000.

[2] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, Order preserving encryption for numeric data, in: Special Interest Group on Management of Data (ACM SIGMOD 2004), June 2004.

[3] M. Acharya, J. Girao, D. Westhoff, Secure comparison of encrypted data in wireless sensor networks, in: 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'05), Trentino, Italy, April 2005.

[4] C. Castelluccia, E. Mykletun, G. Tsudik, Efficient Aggregation of encrypted data in wireless sensor networks, in: 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, San Diego, CA, USA, July 2005.

[5] Wireless medium access control and physical layer specifications for low-rate wireless personal area networks. IEEE Standard, 802.15.4-2003, May2003.

[6] N.B. Chang, M. Liu, Optimal controlled flooding search in a large wireless network, in: 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'05), Trentino, Italy, April 2005.

[7] N.B. Chang, M. Liu, Revisiting the TTL-based controlled flooding search: optimality and randomization, in: 10th Annual International Conference on Mobile Computing and Networking (ACM Mobicom'04), September 2004, Philadelphia, PA, USA, 2004.

[8] D. Dolev, A.C. Yao, On the security of public-key protocols, IEEE Transactions on Information Theory 29 (2) (1983) 198–208.

[9] J. Domingo-Ferrer, A provably secure additive and multiplicative privacy homomorphism, in: Information Security Conference (ISC'02), Springer LNCS, vol. 2433, 2002, pp. 471–483.

[10] J. Girao, D. Westhoff, M. Schneider, CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks, in: IEEE International Conference on Communications (ICC'05), Seoul, Korea, May 2005.

[11] N. Gura, A. Patel, A. Wander, H. Eberle, S.C. Shantz, Comparing elliptic curve cryptography and RSA on 8-bit CPUs, Cryptographic Hardware and Embedded Systems (CHES), Cambridge, MA, USA, 2004, pp. 119–132.

[12] W. Heinzelmann, Application-Specific Protocol Architectures for Wireless Networks, PhD thesis, Massachusetts Institute of Technology, 2000.

[13] J.M. Hellerstein, W. Hong, S.Madden, K. Stanek, Beyond average: Towards sophisticated sensing with queries, in: Workshop on Information Processing in Sensor Networks (IPSN'03), Palo Alto, California, USA, April 2003.

[14] H. Karl, A. Willig, Protocols and Architectures for Wireless Sensor Networks, Wiley, 2005.

[15] S. Madden, M.J. Franklin, J. Hellerstein, W. Hong, TAG: a Tiny aggregation service for ad-hoc sensor networks, in: 5th Symposium on Operating Systems Design and Implementation (OSDI'02), 2002.

[16] A.J. Menezes, P.C. van Oorshot, S.A. Vanstone, Handbook of Applied Cryptography, Discrete Mathematics and its Applications, CRC Press, 1997.

[17] E. Mykletun, J. Girao, D. Westhoff, Re-visited: Public key based homomorphic cryptoschemes for data concealment in wireless sensor networks, in: IEEE International Conference on Communications (ICC'06), Istanbul, Turkey, June 2006.

[18] T. Okamoto, S. Uchiyama, A new Public-Key Cryptosystem as Secure as Factoring, in: Advances in Cryptology – EUROCRYPT'98, 1998, pp. 303–318.

[19] P. Paillier, Trapdooring discrete logarithms on elliptic curves over rings, in: Advances in Cryptology – ASIACRYPT'00, 2000, pp. 573–584.

[20] R.L. Rivest, L. Adleman, M.L. Dertouzos, On data banks and privacy homomorphisms, Foundations on Secure Computation, Academia Press, 1978, pages 169–179.

[21] S. Tilak, N.B. Abu-Ghazaleh, W. Heinzelmann, Collaborative storage management in sensor networks, International Journal of Ad Hoc Ubiquitous computing (2005).

[22] N. Sastry, D. Wagner, Security considerations for IEEE 802.15.4 networks, in: ACM Workshop on Wireless Security (WiSe 2004), co-located with MobiCom 2004 Conference, pp. 32–42.

[23] D. Wagner, Crytanalysis of an algebraic privacy homomorphism (revised version), in: Proceedings of the 6th Information Security Conference (ISC'03), Bristol, UK, October 2003.

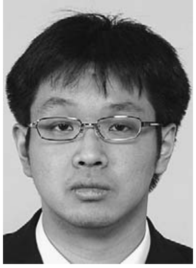
[24] D. Westhoff, J. Girao, M. Acharya, Concealed Data Aggregation for Reverse Multicast Traffic in Wireless Sensor Networks: Encryption, Key Pre-distribution and Routing, to appear as regular paper in IEEE Transactions on Mobile Computing, August 2006.

[25] X. Zeng, R. Bagrodia, M. Gerla, GloMoSim: a library for parallel simulation of large-scale wireless networks, in: Proceedings of the 12th Workshop on Parallel and Distributed Simulation, Banff, Alberta, CA, May 1998.

Joao Girao received his diploma from the University of Aveiro, Portugal, in 2003. Since 2003, he has been a member of the Mobile Internet group at NEC Europe Ltd. in Heidelberg. From 2004 and on, he has been pursuing, in parallel, a PhD with the University of Bochum in the area of security for wireless sensor networks. Other than this area, his research interests include privacy and security issues related to mobility and in ad-hoc networks. He is a student member of both the IEEE and ACM.

Dirk Westhoff holds a PhD from the Distance University of Hagen, Germany. He is a Chief Researcher at NEC Europe Ltd. R&D Network Laboratories in Heidelberg, Germany. He is co-founder of the ESAS (European Workshop on Security in Ad Hoc and Sensor Networks) Workshop series published by Springer. He is involved in the TPC of several ACM and IEEE conferences. His research interests include wireless security, ad hoc and sensor network security, and many other security aspects of distributed mobile communication.

Einar Mykletun is currently a PhD candidate in the School of Information and Computer Science at the University of California, Irvine. His current research interest is focused on providing security in outsourced databases, including the use of secure hardware and techniques to allow an untrusted database server to run SQL queries over encrypted data. This work was done while he was an intern at NEC Europe Network Laboratories.



Toshinori Araki received B.S. and M.E. degrees in computer science in 2003 and 2005 from Tokyo Institute of Technology. He joined Internet Systems Research Laboratories, NEC in 2005. His research interests are cryptography and information security.