A Survey on the Encryption of Convergecast Traffic with In-Network Processing

Steffen Peter, Dirk Westhoff, *Member*, *IEEE*, and Claude Castelluccia

Abstract—We present an overview of end-to-end encryption solutions for convergecast traffic in wireless sensor networks that support in-network processing at forwarding intermediate nodes. Other than hop-by-hop based encryption approaches, aggregator nodes can perform in-network processing on encrypted data. Since it is not required to decrypt the incoming ciphers before aggregating, substantial advantages are 1) neither keys nor plaintext is available at aggregating nodes, 2) the overall energy consumption of the backbone can be reduced, 3) the system is more flexible with respect to changing routes, and finally 4) the overall system security increases. We provide a qualitative comparison of available approaches, point out their strengths, respectively weaknesses, and investigate opportunities for further research.

Index Terms—Cryptography, wireless sensor networks, convergecast, concealed data aggregation.

1 Introduction

Wireless sensor networks (WSNs) are a particular class of ad hoc networks that attract more and more attention both in academia and industry. The sensor nodes themselves are preferably cost-cheap, tiny, and consisting of

1. application-specific sensors,
2. a wireless transceiver,
3. a simple processor, and
4. an energy unit, which may be battery or solar driven.

In particular, we cannot assume a sensor node to comprise a tamper-resistant unit. Such sensor nodes are envisioned to be spread out over a geographical area to form in an indeed self-organizing manner a multihop network. Most frequently, such WSNs are stationary, although mobile WSNs are also conceivable. Potential applications for WSNs—besides military ones—can be found in monitoring environmental data with the objective to understand complex and geographical widespread interdependencies of nature. Examples are the detection of fire in huge forest areas, the monitoring of wildlife animals' movement patterns, or the incremental shift of snow and rocks in the alpine mountains. Further applications for WSNs are envisioned to be on the biomedical sector, public safety, and safety support for vehicles.

One major application scenario for a WSN is to monitor environmental data and to transmit it to a central point. Here, the data are analyzed and eventually serve to initiate some action. Analysis in most scenarios presumes computation of an optimum, e.g., the minimum or maximum, the computation of the average, or the detection of movement pattern. The precomputation of these operations may be either fulfilled at a central point or by the network itself. The latter is beneficial in order to reduce the amount of data to be transmitted over the wireless connection. Since the energy consumption increases linearly with the amount of transmitted data, an aggregation approach helps increase the WSN's overall lifetime. Another way to save energy is to only maintain a connected backbone for forwarding traffic, whereas nodes that perform no forwarding task persist in idle mode until they are reactivated.

It is the aim of this survey to consider WSNs in which messages should be transferred in a confidential way. More precisely, adversaries that eavesdrop communication between the sensors, aggregators, and the sink shall not obtain the exchanged information. This is achieved by encrypting transmitted data. Other security goals, such as integrity, are outside the scope. We assume that adversaries can at least carry out ciphertext-only attacks. However, we will also analyze available solutions according to their protection against more powerful attacks. In principle, there are several possibilities in order to achieve the above security goal. If end-to-end encryption is desired, then applying usual encryption algorithms implies that intermediate nodes have no possibility for efficient aggregation allowing to shrink the size of messages to be forwarded. The application of usual encryption algorithms combined with the requirement of efficient data aggregation provides only the possibility of encrypting the messages hop-by-hop. However, this means that an aggregator has to decrypt each received message, then aggregate the messages according to the corresponding aggregation function and, finally, encrypt the aggregation result before forwarding it. Furthermore, hop-by-hop encryption possesses that intermediate aggregators require keys for decryption and encryption.

It is the contribution of this survey to provide end-to-end encryption for reverse multicast traffic between the sensors

- *S. Peter is with IHP GmBH, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany. E-mail: peter@ihp-microelectronics.com.*
- *D. Westhoff is with NEC Europe Ltd., Kurfürsten-Anlage 36, 69115 Heidelberg, Germany. E-mail: dirk.westhoff@nw.neclab.eu.*
- *C. Castelluccia is with the Institut National de Recherche en Informatique et en Automatique (INRIA) Grenoble - Rhone-Alpes, Inovallee, 655 Avenue de l'Europe Montbonnot, 38334 Saint Ismier Cedex, France. E-mail: Claude.Castelluccia@inrialpes.fr.*

Manuscript received 26 Feb. 2007; revised 18 Dec. 2007; accepted 22 Feb. 2008; published online 20 Mar. 2008.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0025-0207.
Digital Object Identifier no. 10.1109/TDSC.2008.23.

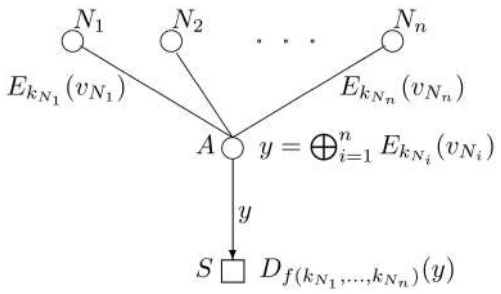

Fig. 1. CDA for WSNs with symmetric PH and multiple secret keys.

and the sink node. We evaluate a set of approaches, which provides aggregators with the possibility to carry out aggregation functions that are applied to ciphertexts. This provides the advantage that intermediate aggregators do not have to carry out costly decryption and encryption operations and, thus, do not require storing sensitive cryptographic keys. The latter ensures an unrestricted aggregator node election process for each epoch during the WSN's lifetime, which is impossible in case of hop-by-hop encryption. Here, only nodes that have stored sensitive key material can act as an aggregator node, and thus, balancing the energy consumption over several nodes is restricted.

In the remainder of this paper, we present a survey of end-to-end encryption solutions with in-network processing, which is known as *Concealed Data Aggregation* (CDA). We outline the main problems that have been solved and present solutions currently available.

2 Basic Principles, Value, and Classification

Before we describe the basic concept of CDA as well as the arising requirements regarding the key management, we introduce a particular encryption transformation named *privacy homomorphic encryption* transformation. A classification of available CDA building blocks completes this section.

2.1 Privacy Homomorphisms

A *privacy homomorphism (PH)* is an encryption transformation that allows direct computation on encrypted data. Let $\mathcal{Q}$ and $\mathcal{R}$ denote two rings, and $+$ and $\oplus$ denote addition operations on the rings. Let $\mathcal{K}$ be the key space. We denote an encryption transformation $E : \mathcal{K} \times \mathcal{Q} \rightarrow \mathcal{R}$ and the corresponding decryption transformation $D : \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{Q}$. Given a, $b \in \mathcal{Q}$ and k, k_1, $k_2 \in \mathcal{K}$, we term

$$a + b = D_k(E_k(a) \oplus E_k(b)) \tag{1}$$

additively homomorphic with a *single secret* key and

$$a + b = D_{f(k_1,k_2)}(E_{k_1}(a) \oplus E_{k_2}(b)) \tag{2}$$

additively homomorphic with *multiple secret* keys. We denote an *asymmetric additively homomorphic* encryption transformation as

$$a + b = D_p\big(E_q(a) \oplus E_q(b)\big) \tag{3}$$

with (p, q) being a *private, public key pair*. The first work on PHs was done in a seminal paper by Rivest et al. [30]. Meanwhile, a set of other candidates, both symmetric and asymmetric, has been proposed as we will see.

2.2 Concealed Data Aggregation

In WSNs, the above introduced PH can be prominently applied for concealing convergecast traffic with simple in-network processing at aggregating intermediate nodes. Such an approach is termed as CDA. We denote $\bigoplus$ as the summing up of $n \geq 2$ encrypted operands with the additive operations $\oplus$. Under such a setting, an aggregator node A is not required to perform decryption and subsequent encryption operations in order to do aggregation operations on the incoming data from sensing nodes $Succ(A) = \{N_1, N_2, \ldots, N_n\}$ with corresponding keys $k_{N_1}, k_{N_2}, \ldots, k_{N_n}$ like it is required when using conventional hop-by-hop encryption (see Fig. 1). This increases the overall system security since there is no lack of security at the aggregating nodes.

Note that CDA, which has originally been proposed in [18], supports various aggregation operations. They are listed in Table 1 with an overview on what needs to be computed at a sensor node, an aggregating node, and a sink node. CDA also supports a hierarchy of aggregating nodes as long as the aggregation function itself supports such a cascaded adjustment. Consequently, the approach is best suited for large-scaled WSNs. Note that depending on what concrete PH [see (1), (2), and (3)] we are applying for the CDA solution, a different *key management* becomes necessary.

2.3 Benefits

Compared to data aggregation with hop-by-hop encryption, we see the substantial advantages of CDA in that

1. neither the encryption keys nor the sensed plaintext information need to be available at aggregating

TABLE 1
Summary of Known Aggregation Functions Using Addition

Function	Sensor node	Aggregator node	Sink node
Average	$E(v_{N_i})$	$\sum_{i=1}^{n} E(v_{N_i}) \| n$	$\frac{Dec(\sum_{i=1}^{n} E(v_{N_i}))}{n}$
Variance	$E(v_{N_i}) \| E(v_{N_i}^2)$	$\sum_{i=1}^{n} E(v_{N_i}) \| \sum_{i=1}^{n} E(v_{N_i}^2) \| n$	$\frac{D(\sum_{i=1}^{n} E(v_{N_i}^2))}{n} - (Average)^2$
Mov. Detect.	$B_i = \{b_n = 0, \cdots, b_i = v_{N_i}, \cdots, b_0 = 0\}$ $E(B_i)$	$\sum_{i=1}^{n} B_i$	$D(\sum_{i=1}^{n} B_i) = \{v_{N_n}, \cdots, v_{N_0}\}$
Checksum	$csum_i = \{1 - \sum_{j=0}^{len(v_{N_i})} v_{Ni,j}\}_{BASE2}$ $E(csum_i)$	$\sum_{i=1}^{n} E(csum_i)$	$csum = D(\sum_{i=1}^{n} E(csum_i))$ $s = \sum_{i=0}^{n} v_{N_i}$ $csum = \{1 - \sum_{j=0}^{len(x)} v_{N_j}\}_{BASE2}$

nodes. This differs from a hop-by-hop encryption approach, where a captured aggregator node would reveal this information.

2. the overall energy consumption of the actual connected backbone can be reduced. For hop-by-hop encryption, each aggregator node needs to first decrypt multiple incoming messages, then aggregate these before encrypting the aggregated data. The CDA approach significantly reduces the energy consumption at aggregator nodes since no encryption and decryption is performed [16]. It is essential to provide overall energy-efficient solutions for the nodes that make up the backbone, since these nodes are most critical for the overall lifetime and connectivity of a WSN.
3. CDA-based end-to-end encryption is much more flexible for varying connected backbones over different epochs. With hop-by-hop encryption, only nodes storing the corresponding key can perform the decryption and thus aggregate data. With CDA, every node can be elected as an aggregator node, since the aggregating nodes do not need to store the key to operate on the incoming ciphertext message. Thus, the election process per epoch is purely based on the remaining energy levels of the nodes. CDA provides confidentiality by not restricting these aggregator-node-election algorithms. This increases the robustness and reliability of the WSN.
4. with CDA, the overall system security level of the WSN increases. Clearly, currently proposed cryptoschemes for WSNs such as RC5, AES, IDEA, or RC4 provide a higher security level and/or require much less execution time compared to any currently available PH. Unfortunately, when applied to WSNs, these schemes run into a security/flexibility tradeoff. With a single networkwide key, the aggregator node election remains as flexible as possible at the cost of almost no security. With group keying or even pairwise keying, the security level of the WSN increases at the cost of almost static routing paths and a fixed set of aggregators in the backbone. The above observation is based on the fact that in systems without tamper-resistant units, the weakest security component is not the cryptoscheme itself but instead the storage policy of sensitive data.

2.4 Classification

We are now in the position to name CDA building blocks and derive criteria for their classification. A classification of the CDA building blocks is depicted in Fig. 2. CDA includes the encryption transformation itself plus a solution for key management. Since most of the available work is addressing key distribution solutions mainly for *unicast traffic*, new approaches are required here.

For the encryption transformation, we categorize solutions regarding whether they satisfy (1), (2), or (3). We further differentiate *deterministic* and *probabilistic* encryption transformations since this impacts the additional requirements of the key management for CDA. Basically on the key management side, we classify *unique keying* where individual keys per sensor node are distributed, *groupwise keying*

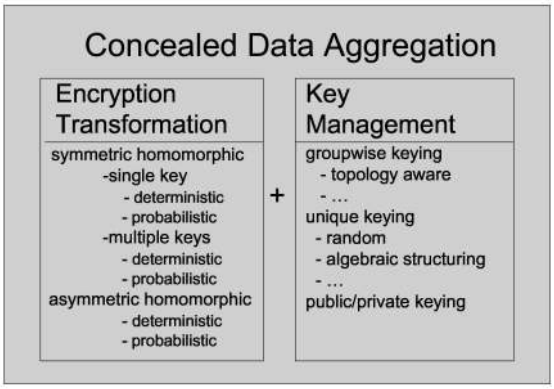

Fig. 2. CDA building blocks and criteria for classification.

where the same secret key is distributed to a subset of nodes, and *public/private keying* in case a PH satisfying (3) is used for CDA. Unique keying can further be subdivided into *random* unique keying and unique keying that supports an *algebraic structuring* of the unique keys within the WSN. Within the class of concepts supporting a groupwise keying, we want to highlight a branch that takes the region of the nodes into account when distributing keys. To the best of our knowledge, this classification reflects all the key management concepts that are currently available for CDA.

3 Attack Scenarios

Although many of the following attacks can be repelled by protocols and technologies other than cryptographic algorithms (e.g., secure routing, safe infrastructure), we focus on the resistance of the actual CDA scheme. The potential targets of an adversary are the deduction of

1. the secret key (total break of the system),
2. plaintexts not previously known (corresponds to the classical unauthorized decryption), and
3. additional ciphertexts (usually used to forge malicious ciphertexts).

Obviously, the revealing of the secret key, i.e., the total break, is the worst case scenario. It allows the attacker to decrypt and encrypt every message in the system. The deduction of plaintexts usually from transmitted ciphertexts compromises the secrecy. In contrast, the deduction of additional ciphertexts can imply a loss of any trust in the network, since every received message can be forged by an adversary.

3.1 Passive Attacks

Passive attacks comprise all attacks that do not require the adversary to actively interfere with the connection. In order to perform such an attack, the adversary needs to do nothing but listen to transmitted packets. The eavesdropped information can be evaluated and usually cryptoanalyzed in order to obtain secret information. Though the worst case result of such an attack would be the deduction of the secret key, most attacks aim at revealing the plaintexts or at gathering information for further actions.

Passive attacks can be performed relatively easy. Given the characteristics of the broadcast medium those attacks

are not detectable, which make them highly dangerous. It must be the primary security goal of a cryptoscheme that an adversary is not able to gain any information by simple eavesdropping. With regard to these properties, the requirements are very similar to classic cryptoschemes.

3.1.1 Ciphertext Analysis

A very common and actually the most basic attack is the analysis of encrypted packets. In such an analysis, the adversary wants to obtain information only by interpreting ciphertexts. A secure cryptographic system must ensure that it is not possible to gain any inappropriate information (plaintext, key, statistical information). Additionally, it must be provided that an attacker cannot decide whether an encrypted packet corresponds to a specific plaintext or not. In particular, in WSNs with a scarce domain of values, the latter attack can very efficiently result in a deduction of the plaintexts.

3.1.2 Known Plaintext Attack

In this kind of attack, the adversary tries to determine secret information with the additional knowledge of plaintexts. With known plaintext and corresponding ciphertext, it is the aim of the adversary either to reveal the secret key or at least to gather additional information that can be exploited to deduct malicious ciphertexts or decrypt other messages.

In a WSN scenario, such an attack is very likely since an adversary can obtain plaintexts corresponding to the ciphertexts that are sent via the air on various ways, e.g., by

- guessing the values of the plaintext (e.g., temperature),
- by an own sensor that determines the plaintext values,
- physically accessing the deployed sensor, or
- manipulating the sensor readings (e.g., heat the sensor).

Assuming the cryptoscheme uses the same secret key on every node (see (1)) this sort of attacks is a serious threat if the scheme does not provide resistance. Resistance to known plaintext attacks means that, even with a large set of corresponding plain- and ciphertexts, it is not possible to deduce secret keys or additional cipher- or plaintexts out of the known set.

In case of a deterministic cryptoscheme, recorded database of ciphertexts for every possible plaintext destroys any security. In particular for WSNs with a scarce domain of sensed or transmitted values, such a straightforward attack is not only considerable but also very threatening.

3.2 Active Attacks

The described passive attacks do not require the adversary to actively interfere the communication. In case of active attacks, the adversary is assumed to be able to perform such interferences, i.e., to catch, destroy, modify, and send packets. An attacker could catch a packet, analyze it, modify the content, and even replace the original packet in the network. Such attacks require the attacker to have a lot more knowledge and technical instruments. Though such attacks are much more complicated and expensive than passive attacks, their potential damage can be also much more severe. A successful attack that allows an adversary to change or forge any packet can render the whole network useless. In such a case, every received packet could be malicious so that every sensed value and every action could have been modified in the interest of an adversary. As we will see also (or especially), CDA algorithms are vulnerable to active attacks.

3.2.1 Replay Attacks

Replay attacks are the easiest variation of active attacks. Valid packets that have been sent before are transmitted later in order to achieve a malicious effect. For CDA in WSNs, it is considerable to record the ciphertext in a situation where the plaintext is known or causes a noticeable specific reaction of the system. The recorded packet can be resent later in order to initiate a desired action of the system or to pretend a situation that is not actually sensed.

For example, in a movement detection scenario, a trespasser can keep sending the previously recorded "no movement" signal while he is moving in the protected area. The system receives the correctly encoded messages and does not trigger the alarm. Another variation of this attack is not to replay a previously sent message but to replay the messages of a different node in order to cover that one node is either disabled or would sense undesired values. This way a recorded "movement detected" signal could be replicated on every node in the system so that the actual intrusion cannot be detected in time.

Though there are several possible countermeasures that are based on protocols (e.g., time stamps, node ID), it would be desirable to have a resistance to this kind of attack in the initial CDA algorithm. It means that it is not possible to take a correctly encoded message recorded at different time or another place without being noticed by the decryption algorithm.

3.2.2 Malleability

The idea of this very dangerous attack is to alter the content of a valid encrypted packet without leaving marks. A simple variation of that attack would be randomly generated ciphertexts that are syntactically correct. In such case, the adversary does not know the actual effect of the modification, but its intention is to harm the system. A more sophisticated variation is a specific alternation of a ciphertext. For example, the adversary knows that a sensor transmits the current temperature of about 20 °C and he wants to increase the encrypted value to 40 °C. For some PH schemes, it is possible to alter the content (i.e., the plaintext) of an encrypted packet without knowing the concrete content. Here, the attacker can increase the transmitted temperature by 20 °C even without being able to decrypt the original message. Due to their algebraic properties, PH schemes may be very vulnerable to this kind of malleability.

Additionally for CDA algorithms, that weakness becomes more severe because the decrypting unit receives only a derivation (i.e., the aggregate) of the sensed values. This means that the modified value is aggregated several times before it finally will be decrypted. Consequently, possible marks of the modification can be blurred and even if the sink node realizes the modification, it does not know

the source of it. Additionally, it can be assumed that most approaches of *Resilient Data Aggregation (RDA)* are not applicable to CDA networks because only the sink node becomes aware of the content. Thus, aggregating nodes cannot detect unreasonable content, and even worse with the aggregation the modification will be covered.

3.2.3 Unauthorized Aggregation

Actually, the unauthorized aggregation is a variation of the malicious modification. However, since it is a very specific weakness of PH schemes, we will treat them separately. The idea of such an attack is to take two or more proper ciphertexts and aggregate them in order to inject the result somewhere into the network. Like the normal aggregation nodes, the attacking aggregator does not need to know the plaintexts of the individual messages in order to aggregate them to a properly looking ciphertext. An adversary could use that property to vandalize the system, but it is also considerable to apply it more specifically. With known or assumed ciphertext/plaintext combinations, an attacker can modify packets that are well directed. For example, an adversary knows the ciphertext $C1$ for the temperature of about 20 °C. In order to increase the current sensed temperature with the ciphertext $C2$ by 40 °C, he could aggregate $C3 = C2 + C1 + C1$ and replace $C2$ with $C3$.

There are two considerable ways of protecting a PH scheme from unauthorized aggregation. First, the aggregation may need a secret key in order to be performed. Thus, an adversary cannot execute the aggregation without knowing or breaking the secret key. The second approach would be to ensure that every ciphertext cannot be used more than once so that the decryption unit can detect the unauthorized aggregation.

3.2.4 Forge Packets

An adversary does not need to modify existing packets if she is able to create properly encoded ciphertexts with a specific content. The attacker could simply substitute the packet of the actually sensed value with the forged one. If there is no protection to this issue, the receiving unit can never be sure whether the received packet was really sensed. It can be assumed that every public key approach, where a public key is used to encrypt the plaintexts, is initially vulnerable to this attack. A PH scheme that is resistant to maliciously forged packets must not allow any third party to create properly encoded messages at least not without being able to detect the interference during decryption.

3.3 Physical Attacks

Physical attacks as they are meant here embrace attacks against the hardware of the node. In the context of PH schemes, it does not include the attack of disabling a node, because this would not implicitly be a threat against the security of the cryptoscheme. A serious threat is the capturing of nodes. The access to the flash and the memory may reveal key information that can compromise the entire network. In particular, symmetric encryption schemes that use the same key on every node are vulnerable. A captured and completely revealed node with all its key information corresponds to a total break of the network. A captured node could also be a problem because it allows the adversary to collect a set of plaintext/ciphertext pairs with known plaintexts. It could be the basis for further actions, e.g., known plaintext attacks. It would be desirable that under any circumstances a captured node cannot turn out to be a threat for the rest of the system, i.e., it is not possible to extract information that could be applied in a further attack.

4 Requirements Analysis

Prior to presenting available solutions for CDA, we will outline the criteria and desired design requirements of an appropriate CDA solution. Beneficial requirements regarding the security of the system are given as follows:

- *Provable Security:* The security level of the encryption scheme should be measurable and it should be based upon the commonly agreed hardness of a mathematical problem to be provably computationally secure.[1]
- *Sensor Compromise:* The compromise of a subset of sensor nodes should not assist in revealing aggregated data.
- *System Security:* From the two points mentioned above, we can define the overall system security as being the weakest of the two.
- *Key Management:* The key management should be kept simple enough to avoid bandwidth intensive techniques needed to identify the encryption keys being used by sensors.
- *Ciphertext Expansion:* The expansion in bit size attributed to encryption should be moderate.
- *Probabilistic Encryption:* Encryption of the same plaintext should not, with high probability, yield the same ciphertext.

In addition to the security requirements, the design space for an appropriate CDA approach should also consider requirements regarding the lifetime, flexibility, and robustness of the system:

- *Efficient Computations:* Cryptographic operations performed at sensors should not be overly expensive.
- *Aggregator Node Election:* The algorithm for electing aggregator nodes should not need to take into account security parameters, thereby allowing it to make selections purely based on lower layers' parameter, e.g., the remaining energy level of the nodes.
- *Network Topology:* Each sensor node is aware of its aggregator node and each aggregator node knows its reporting sensor nodes. If a node changes its group, it is considered to be announced in the network.

Note that the second *security* criterion rules out a hop-by-hop encryption approach, as the compromise of a few nodes may be enough to render the WSN insecure. While the third point reveals the weaknesses of symmetric key schemes according to (1) in WSN settings when assuming non-tamper-resistant sensors. Probabilistic encryption proves

1. A cryptoscheme is said to be computationally secure if the cost of an attack outweighs the value of the encrypted data.

useful to avoid divulging information from ciphertexts only, as identical environment values may often be measured and encrypted by neighboring nodes. We note that non-tamper-resistant sensor nodes can be compromised and have their contents revealed by an attacker (such as public keys and current unencrypted measurements). However, it should not be possible to learn (encrypted) aggregated values from the compromise of a single node or a minor fraction of the WSN. The *lifetime* criteria relate to the lifetime of nodes, as computations and especially communication are energy intensive. By performing in-network aggregation, nodes avoid having to forward every received packet toward the reader, thereby drastically reducing the overall bandwidth consumption.

Next, we will examine potential cryptoscheme candidates that meet some of the desired criteria outlined in this section. None of the discussed candidates meets all the desired criteria.

5 Encryption Transformations

5.1 Symmetric Homomorphic Encryption Transformations

Symmetric PH schemes require identical secret information for encryption and decryption. In this section, we present four schemes that have the additive PH property and promise to be suitable for the application in WSNs.

5.1.1 Domingo-Ferrer Scheme

In [11], Domingo-Ferrer introduced a symmetric PH scheme (DF) that has been proposed as efficient PH cryptographic system for WSNs in [16]. The PH is *probabilistic*, which means that the encryption transformation involves some randomness that chooses the ciphertext corresponding to a given cleartext from a set of possible ciphertexts.

Domingo-Ferrer (DF) algorithm [11]

Parameter: *public key: integer* $d \geq 2$, *large integer* M
secret key: $k = (r, g)$
small g *that divides* M; r *so that* r^{-1} *exists in* $\mathbb{Z}_M$

Encryption: split m into d parts $m_1 \dots m_d$ that
$\sum_{i=1}^{d}(m_i) \bmod g = m$
$C = [c_1, \dots, c_d] = [m_1 r \bmod M, m_2 r^2 \bmod M, \dots, m_d r^d \bmod M]$

Decryption: $m = (c_1 r^{-1} + c_2 r^{-2} + \dots + c_d r^{-d}) \bmod g$

Aggregation: Scalar addition modulo M
$C12 = C1 + C2 = [(c1_1 + c2_1) \bmod M, \dots, (c1_d + c2_d) \bmod M]$

The set of cleartext is $\mathbb{Z}_g$, and the set of ciphertext is $(\mathbb{Z}_M)^d$.

DF has both the additive and the multiplicative PH properties. For the ciphertext multiplication, all terms are cross-multiplied in $\mathbb{Z}_g$, with the d_1-degree term by a d_2-degree term yielding a $(d_1 + d_2)$-degree term. Terms having the same degree are added up.

DF is a symmetric algorithm that requires the same secret key for encryption and decryption. The aggregation is performed with a key that can be publicly known, i.e., the aggregator nodes do not need to be able to decrypt the encrypted messages. However, it is required that the same secret key is applied on every node in the network that needs to encrypt data. The message size is $d \cdot n$ bit. For very secure parameter combinations ($d > 100$), the messages become very big [31]. However, Girao et al. [16] showed that with reasonable parameters it also fits the needs of constrained devices.

5.1.2 Castelluccia-Mykletun-Tsudik Scheme

Castelluccia, Mykletun, and Tsudik [7] propose a simple and provably secure additively homomorphic stream cipher that allows efficient aggregation of encrypted data. The main idea of the scheme is to replace the exclusive-OR (XOR) operation typically found in stream ciphers with modular addition $(+)$. Since this new cipher only uses modular additions (with very small moduli), it is very well suited for CPU-constrained devices.

Castelluccia, Mykletun, Tsudik (CaMyTs) algorithm [7]

Parameter: *select large integer* M

Encryption: *Message* $m \in [0, M-1]$,
randomly generated keystream $k \in [0, M-1]$
$c = (m + k) \bmod M$

Decryption: $Dec(c, k, M) = c - k (\bmod M)$

Aggregation: Let $c_1 = Enc(m_1, k_1, M)$ and
$c_2 = Enc(m_2, k_2, M)$
For $k = k_1 + k_2$, $Dec(c_1 + c_2, k, M) = m_1 + m_2$

It is assumed that $0 \leq m < M$. Due to the commutative property of addition, the above scheme is additively homomorphic. In fact, if $c_1 = Enc(m_1, k_1, M)$ and $c_2 = Enc(m_2, k_2, M)$, then $c_1 + c_2 = Enc(m_1 + m_2, k_1 + k_2, M)$.

Note that if n different ciphers c_i are added, then M must be larger than $\sum_{i=1}^{n} m_i$; otherwise, *correctness* is not provided. In fact, if $\sum_{i=1}^{n} m_i$ is larger than M, decryption will result in a value m' that is smaller than M. In practice, if $p = \max(m_i)$, then M should be selected as $M = 2^{\lceil \log_2(p*n) \rceil}$.

The keystream k can be generated by using a stream cipher, such as RC4, keyed with a node's secret key s_i and a unique message ID. This secret key is precomputed and shared between the node and the sink, while the message ID can either be included in the query from the sink or it can be derived from the time period in which the node is sending its values in (assuming some form of synchronization).

5.1.3 Authenticated Interleaved Encryption-Based Scheme

One limitation of the previous proposal is that the identities of the nonresponding nodes (or responding nodes, whichever is expected to be smaller) need to be sent along with the aggregate to the sink. If the network is unreliable, this can represent an important overhead and scalability problem. It is therefore important to devise methods for reducing this cost.

In *Authenticated Interleaved Encryption (AIE)* [8], or, very similar, in [23], each node shares a pairwise key with its direct parent, its two-hop parent, three-hop parent, ..., and n-hop parent, where n is a system parameter. These keys can be established using a scheme such as [13].

When a sensor, N_i, sends a message, it encrypts it n-time using the additively homomorphic scheme described in [7]. The first time with the key it shares with its direct parent,

the second time with the key it shares with its two-hop parent, ..., the nth time with the key its shares with its n-hop parent. The sensor sends the result c_i to its parent along with its identifier.

An aggregator A_i adds up all the ciphers c_l it receives from all of its direct children. It then decrypts the results using the sum of the pairwise keys it shares with each of its direct children, two-hop children, ..., n-hop children. The result is then encrypted n times with the keys it shares with its parent, two-hop parent, ..., and nth. The result is forwarded to A_i's parent along with the identifiers of the children that have contributed to the resulting cipher. The messages get then securely aggregated hop-by-hop until the sink.

With the AIE scheme, each aggregator has to forward at most $\sum_{i=1}^{n-1} d^i$ identities, where d is the degree of the tree, i.e., number of children per node. This is much less than the CaMyTs scheme. In the original CaMyTs scheme, the number of identities to be forwarded increases as the aggregated message gets closer to the root. At the level h of the tree ($h = 0$ being the leaves), $O(d^h)$ identities have to be forwarded by each aggregator. If the aggregator tree has many levels, this can become problematic. In contrast, with AIE, the number of identities to be forwarded is bounded and only depends on the parameters n and d, where d is smaller than h.

Note, however, that the AIE-based scheme is less secure than the original scheme. An attacker that corrupts n consecutive nodes can actually retrieve the aggregated value at the lowest corrupted aggregator in the tree. With the original scheme, corrupting aggregators does not reveal any information about the aggregated value.

There is a clear trade-off between the number of identities to be forwarded (i.e., bandwidth cost) and security. By decreasing n, the bandwidth cost decreases but so does the security. By increasing n, the bandwidth cost and security increase. If $n = 1$, the AIE scheme is similar to hop-by-hop encryption. This configuration is optimal in terms of bandwidth but very weak security-wise. On the other hand, if $n = h$ (where h is the number of level in the tree), the AIE scheme is similar to the original aggregation scheme in [7]. Its bandwidth cost is high, but its security is maximum.

5.1.4 Hybrid Symmetric PH Approach

In [26], an approach has been proposed that combines two known PH algorithms. It is the notion to increase the security and cope with security issues of single PHs by performing cascaded encryptions. The idea of this action is to combine the advantages of both cryptoschemes. Considering that one scheme is vulnerable to one attack and another scheme has another weakness, the combined algorithm can cover both issues.

Considering we have two PH encryption transformations $E_1 : \mathcal{K}_1 \times \mathcal{Q}_1 \rightarrow \mathcal{R}_1$ and $E_2 : \mathcal{K}_2 \times \mathcal{Q}_2 \rightarrow \mathcal{R}_2$ with corresponding decryption and properties as described in Section 2. A cascaded PH is the successively performed execution of both encryption functions that results in the transformation $E_C : \mathcal{K}_2 \times \mathcal{K}_1 \times \mathcal{Q}_1 \rightarrow \mathcal{R}_2$ sustaining the homomorphic property:

$$E_{K2}(E_{K1}(a)) \oplus E_{K2}(E_{K1}(b)) = E_{K2}(E_{K1}(a+b))$$

and

$$a + b = D_{K1}(D_{K2}(E_{K2}(E_{K1}(a)) \oplus E_{K2}(E_{K1}(b)))).$$

E_{K1} stands for the inner cryptographic algorithm and E_{K2} for the outer one. This means that the plaintext a is encrypted with algorithm E_1 and the resulting ciphertext is encrypted again with algorithm E_2, while preserving the homomorphic property corresponding to the algebraic operation $+$.

Such a chain has some requirements on the encryption transformation: both encryption schemes must be additive PH, and the ranges of results of inner encryption E_1 must fit to the domain of E_2, i.e., $\mathcal{R}_1 = \mathcal{Q}_2$.

As an example, the combination of CaMyTs and DF is demonstrated. As we will see in Section 7, this combination results in a very secure CDA approach that is still suitable to lightweight devices.

The DF/CaMyTs combination is algebraically sound since CaMyTs as E_1 encryption maps the plaintexts that are in $\mathbb{Z}_n$ $E_1 : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$, and DF uses the resulting ciphertexts for its encryption $E_2 : \mathbb{Z}_n \rightarrow \mathbb{C}$, while $\mathbb{C}$ is a usual known DF ciphertext.

It should be mentioned that an aggregation operation for DF/CaMyTs performed on an aggregation node requires exactly the same effort as for the standalone DF. It is not necessary to consider the embedded CaMyTs encryption. Since most security concerns are already covered by CaMyTs, the DF parameters, especially the setting parameter d, do not need to be too big. Thus, the potential ciphertext expansion is moderate.

However, with both encryption methods, there are indeed the technical problems of both approaches. With $d > 1$, the encrypted message size increases and there is still the ID issue to indicate nonresponding nodes.

CaMyTs + Domingo-Ferrer (CaMyTs/DF) algorithm [26]

Parameter: *public key: large integer* M, $d \geq 2$
secret key: g *that divides* M; r *so that* $r^{-}1$ exists in $\mathbb{Z}_M$

Encryption: *randomly generated keystream* $k \in [0, M-1]$
$e_1 = (k + m) \bmod M$
split e_1 into d parts $m_1 \ldots m_d$ that $\sum_{i=1}^{d}(m_i) \bmod g = e_1$ $C = [c_1, \ldots, c_d] = [m_1 r \bmod M, m_2 r^2 \bmod M, \ldots, m_d r^d \bmod M]$

Aggregation: scalar addition modulo M (like DF)

Decryption: $d_1 = (c_1 r^{-1} + \ldots + c_d r^{-d}) \bmod g$
$m = (d_1 - k) \bmod M$
where k is the sum of aggregated keystreams

5.2 Asymmetric Homomorphic Encryption Transformations

In light of inevitable problems connected with key distribution and synchronization required for symmetric encryption schemes, we are encouraged to revisit the use of public key encryption schemes that

1. are additively homomorphic (allowing for in-network aggregation of particular aggregation functions),

2. exert the required security levels,
3. involve relatively cheap computations,
4. are probabilistic,
5. produce relatively short ciphertexts, and
6. by nature of public key methodology, require no sensitive key material to be stored at encrypting sensors.

We are especially interested in the use of elliptic curve cryptoschemes, due to 1) their use of small keys, which leads to short ciphertexts, 2) the smaller real estate required for hardware implementations (number of gates), and 3) a better security-per-bit ratio.

A large subgroup of asymmetric PHs is the family of high degree residuosity class-based cryptographic algorithms, for example from Paillier [24], Benaloh [5], Naccache and Stern [22], and Okamoto and Uchiyama (OU) [29]. Though all these public key schemes provide the additive PH, we only describe the latter one, since all algorithms exploit similar mathematical problems. Additionally, due to their long keys that imply large messages and high computation efforts, the application of these schemes in a WSN scenario is at least questionable, while the OU scheme shows the most promising results [21].

5.2.1 Okamoto-Uchiyama Scheme

In Eurocrypt '98, Okamoto and Uchiyama proposed a new public-key cryptosystem (OU) as secure as factoring and based on the ability of computing discrete logarithms in a particular subgroup [29]. Their scheme is characterized by probabilistic encryption, additive homomorphic properties, and relating the computational complexity of the encryption function to the size of the plaintext.

Specifically, for an odd prime p, the p-Sylow subgroup is defined as $\gamma_p = \{x < p^2 \mid x = 1 (\mathrm{mod}\ p)\}$, and $|\gamma_p| = p$. A function L that maps elements from γ_p to $\mathbb{Z}_p$ is defined as $L(x) = (x-1)/p$. Function L has homomorphic properties from multiplication to addition. For elements $a, b \in \gamma_p$, $L(a * b) = L(a) + L(b) (\mathrm{mod}\ p)$, and for $c \in \mathbb{Z}_p$, $L(a^c) = c * L(a)$.

Now, let p and q be random k-bit primes and set $n = p^2 q$. For an n of approximately 1,024 bits, a choice of k could be 341. Next, randomly choose a $g \in_R \mathbb{Z}_n$ such that element $g_p = g^{p-1} (\mathrm{mod}\ p^2)$ has order p. Finally, set $h = g^n (\mathrm{mod}\ n)$. The additive homomorphic property is achieved through the multiplication of ciphertexts: $Enc(m_1 + m_2) = Enc(m_1) \times Enc(m_2)$.

Okamoto-Uchiyama (OU) algorithm [29]
Parameter: *public key:* $n = p^2 q, g, h$
Private key: (p, q)
Encryption: plaintext $m \in 2^k$,
$r \in_R \mathbb{Z}_n$,
ciphertext $c = g^m h^r (\mathrm{mod}\ n)$
Decryption: $c' = c^{p-1} (\mathrm{mod}\ p^2)$
compute $m = L(c') L(g_p)^{-1} (\mathrm{mod}\ p)$
Note that $c^{p-1} (\mathrm{mod}\ p^2) = g^{m(p-1)} g^{nr(p-1)} = g_p^m (\mathrm{mod}\ p^2)$

5.2.2 ECC Schemes Suggested by Paillier

In [25], Paillier describes three new probabilistic encryption schemes that use elliptic curves over rings and exhibit additive homomorphic properties. All three schemes are elliptic curve variants of the previously described public key encryption algorithms, namely those proposed by Naccache and Stern [22], Okamoto and Uchiyama [29], and Paillier [24]. Since the encryption schemes in [25] are implemented over elliptic curves and meet some of our desired criteria, we describe each and investigate their applicability for aggregation in WSNs. Common to each scheme is that the elliptic curve is defined over $\mathbb{Z}_n$ or $\mathbb{Z}_n^2$, where n is the product of large primes, and that they are provably secure against chosen plaintext attacks. All three schemes provide additive homomorphic capabilities through the summation of ciphertexts. The reader is referred to [25] for more detailed descriptions of the cryptosystems.

Elliptic Curve Okamoto-Uchiyama Encryption. The Elliptic Curve Okamoto-Uchiyama (EC-OU) Encryption uses the fact that discrete logarithms are easy to compute in curves $E_p(\overline{a_p}, \overline{b_p})$ over F_p, which have trace of Frobenius one (anomalous curves)[2], where values $\overline{a_p}$, $\overline{b_p}$ denote a particular curve. Paillier extends this discrete logarithm recover ability property to a p-subgroup of $E_{p^2}(a, b)$ such that the projection onto F_p gives the twist of an anomalous curve.

Define $n = p^2 q$, where p, q are large 341-bit primes, and $p \equiv 2 (\mathrm{mod}\ 3)$. Values $\overline{a_p}$, $\overline{b_p} \in F_p$ are chosen such that $E_p(\overline{a_p}, \overline{b_p})$ is of order $p + 2$. A random curve $E_q(\overline{a_q}, \overline{b_q})$ along with a lift $E_{p^2}(a_p, b_p)$ of $E_p((\overline{a_p}, \overline{b_p})$ to F_{p^2} is chosen. Then, by using the Chinese Remainder Theorem (CRT), $E_{p^2}(a_p, b_p)$ and $E_q(\overline{a_q}, \overline{b_q})$ are combined to get the curve $E_n = E_n(a, b)$, where $a, b \in \mathbb{Z}_n$. A base point $G \in E_n$ of maximal order $lcm(|E_{p^2}|, |E_q|)$ is chosen and $H = nG$. The cryptosystem's security can be shown equivalent to factoring $n = p^2 q$.

Elliptic Curve Okamoto-Uchiyama (EC-OU) algorithm
Parameter: *Public key:* $n = p^2 q, G, H, E_n$
Private key: p
Encryption: plaintext $m < 2^{k-1}$,
$r \in_R 2^{2k}$,
ciphertext $C = mG + rH$
Decryption: compute $m = \frac{\psi_p((p+2)C)}{\psi_p((p+2)G)} (\mathrm{mod}\ p)$
where $\psi_p(x, y) = -\frac{x}{y} (\mathrm{mod}\ p^2)$ and has the property
that if $P = mG$ for arbitrary points P, G, then
$m = \frac{\psi_p(P)}{\psi_p(G)} (\mathrm{mod}\ p)$
provided that $G \neq \mathcal{O}_{p^2}$.

Elliptic Curve Naccache-Stern Encryption. Elliptic Curve Naccache-Stern Encryption (EC-NS) is constructed in a manner similar to KMOV [19], whereby factoring-based algorithms are exported to particular families of elliptic curves.[3] The applicable curves have the following specific form:

$$E_n(0, b) : y^2 = x^3 + b (\mathrm{mod}\ n), \text{ for } b \in \mathbb{Z}_n^*$$

2. Specifically, such a computation of discrete logarithms requires $O(\log^3 p)$-bit operations.

3. The KMOV paper introduced elliptic curve schemes that, like the RSA cryptosystem, base their security of the difficulty of factoring a value $n = pq$, where p and q are large primes. This differs from typical ECC solutions that base themselves on the computationally hard discrete logarithm problem.

with $p \equiv q \equiv 2 (\mathrm{mod}\ 3)$ and $\mu = |E_n(0,b)| = lcm(p+1, q+1)$. Further requirements are given as follows:

$$p+1 = 6 \times u \times p', \text{ where } u = \prod p_i^{\delta_i}$$
$$q+1 = 6 \times u \times q', \text{ where } u = \prod q_i^{\delta_j}$$

for $\mathcal{B}$-smooth integers[4] u and v of (roughly) equal bit size such that $gcd(6, u, v, p', q') = 1$, primes p', q', and $B = O(\log n)$. By the properties of primes p and q, the two curves $E_p(0,b)$ and $E_q(0,b)$ are cyclic groups of orders $p+1$ and $q+1$, respectively.

Let G be a (base) point of $E_n(0,b)$ such that its order is a multiple of $\sigma = uv$. Then, encryption of a plaintext $m \in \mathbb{Z}_\sigma$ can be realized as

$$Enc(m) = C = (m + \sigma r)G, \text{ where } r \in_R E_n(0,b).$$

Because σ is $\mathcal{B}$-smooth, it is possible to efficiently compute discrete logarithms for a base of degree σ by using a combination of the baby-step giant-step and Pohlig-Hellman algorithms [20]. Thus, with the knowledge of μ, decryption can be accomplished by computing the discrete logarithm of $(\mu/\sigma)C$ with respect to the base $G' = (\mu/\sigma)G$. The security of the scheme is equivalent to computing residue classes on $E_n(0,b)$. The following outlines the cryptosystem:

Elliptic Curve Naccache-Stern (EC-NS) algorithm

Parameter: *Public key:* $n = pq, b, \sigma, G, E_{n(0,b)}$
Private key: (p,q) or $\mu = lcm(p+1, q+1)$
Encryption: plaintext $m \in \mathbb{Z}_\sigma$,
$r \in_R \mathbb{Z}_n$,
ciphertext $C = (m + \sigma r)G$
Decryption: compute $u = (\mu/\sigma)C = mG'$.
Use Pohlig-Hellman and baby-step giant-step to compute the discrete log of u in base G'

Elliptic Curve Paillier Encryption. The cryptoscheme Elliptic Curve Paillier (EC-P) extends the settings of EC-OU to curves defined over $\mathbb{Z}_{n^2}$, where $n = pq$ and p, q are large primes with the properties that $p \equiv q \equiv 2 (\mathrm{mod}\ 3)$. Values $\overline{a_p}, \overline{b_p} \in F_p$ and $\overline{a_q}, \overline{b_q} \in F_q$ are chosen such that $E_p(\overline{a_p}, \overline{b_p})$ is of order $p+2$ and $E_q(\overline{a_q}, \overline{b_q})$ is of order $q+2$. The lifted curves $E_{p^2}(\overline{a_p}, \overline{b_p})$ and $E_{q^2}(\overline{a_q}, \overline{b_q})$ are chosen and combined to get $E_{n^2}(a,b)$. A base point $G \in E_{n^2}$ of order divisible by n is chosen, possibly of maximal order $n\mu$, where $\mu = \mu(n) = lcm(p+2, q+2)$. The security of EC-P is based upon the problem of computing residuosity classes over E_{n^2}. Here is the scheme:

Elliptic Curve Paillier (EC-P) algorithm

Parameter: *Public key:* $n = pq, G, E_{n^2}$
Private key: $\mu = lcm(p+2, q+2)$ or equivalently (p,q)
Encryption: plaintext $m \in \mathbb{Z}_m$,
$r \in_R \mathbb{Z}_n$,
ciphertext $C = (m + nr)$
Decryption: compute $m = \frac{\psi_n(\mu C)}{\psi_n(\mu G)} (\mathrm{mod}\ n)$

4. An integer is said to be $\mathcal{B}$-smooth if all its prime factors are $\leq \mathcal{B}$.

It is important to note that in [14], Galbraith shows that the use of anomalous curves in the way it is described in the two schemes above is insecure. The attack reveals the private key by efficiently extracting it from the public key. Although the same author proposes a variation of the EC-P scheme, this new scheme is not as efficient and, therefore, requires too much computation for the scenarios we are considering.

5.2.3 Elliptic Curve ElGamal Encryption Scheme

A very different cryptoscheme working on elliptic curves is the elliptic curve ElGamal encryption scheme (EC-EG). It is equivalent to the original ElGamal scheme [12] but transformed to an additive group. Key setup consists of choosing an elliptic curve E together with a prime p and generator G. Its security is based upon the Elliptic Curve Discrete Log Problem (ECDLP).

Elliptic Curve ElGamal (EC-EG) algorithm [21]

Parameter: *Public key:* E, p, G, $Y = xG$, where $G, Y \in F_p$
Private key: $x \in F_p$
Encryption: plaintext $M = map(m)$,
$k \in F_p$,
ciphertext $C = (R, S)$, where
$R = kG$, $S = M + kY$
Decryption: $M = -xR + S = -xkG + M + xkG$,
$m = rmap(M)$

EC-EG is additively homomorphic, and ciphertexts are combined through addition. The summation of two EC-EG ciphertexts requires two point additions, namely one for each of the ciphertext components R and S.

map() refers to the mapping function used to map values (e.g., plaintexts) into points on the curve and vice versa. This mapping needs to be deterministic such that the same plaintext always maps to the same point. Additionally, the function needs the following property to hold: for all $a_1, a_2 \in F_p$, $map(a_1 + a_2) = map(a_1) + map(a_2)$. An applicable homomorphic mapping function is proposed by VoteHere in [2] and is based upon using multiples of a generator element to represent mapped values. The 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. However, the demapping of the mapped point jG back to j is not trivial. Since it is the fundamental property of ECC that the point multiplication is not efficiently invertible, the only solution is a brute force computation that relies on a limited domain of the mapping. In most cases, this approach is very reasonable.

5.2.4 Comparison of Asymmetric Schemes

Table 2 compares the performance of the described public key homomorphic encryption candidates applying the results from [21]. All table entries consist of two values: 1) the formulas used to determine the respective costs and 2) the actual number of computations and bits transmitted when applying the formulas to our set of assumed values, as described below. The formulas refer to parameters of the respective schemes, i.e., the p in EC-EG refers to the 163-bit

TABLE 2
Performance Comparison of Candidates: 1) Formulas and 2) Number of Computations (1,024-Bit Modular Multiplications) and Bandwidth (Number of Bits)

Scheme	Encryption		Addition		Decryption		Bandwidth	
EC-NS	$\frac{15}{2}\|n\|$	1800	$5\|n\|$	5	$\frac{15}{2}\|n\|$	7680	$\|n\|+2$	1026
EC-OU	$5+\frac{15}{2}\|m\|+\frac{15}{2}\|2k\|$	690	$5\|n\|$	5	$\frac{15}{2}\|p\|$	2558	$\|n\|+2$	1026
EC-P	$\frac{15}{2}\|n^2\|$	33105	5	20	14	56	$\|n^2\|+4$	2052
EC-EG	$2\frac{15}{2}\|p\|+\frac{15}{2}\|m\|$	38	$10\|p\|$	1	$\frac{15}{2}\|p\|+5$ + map	97	$2(\|p\|+1)$	328
OU	$\frac{3}{2}(\|m\|+\|r\|)+1$	132	$1\|n\|$	1	$\frac{3}{2}\|p\|$	512	$\|n\|$	1024

modulus defining the elliptic curve, while the p in EC-OU is the (typically) 341-bit prime that is used to construct the modulus n. All computations (the second part of the entries) are converted to and measured in terms the number of base units (1,024-bit modular multiplications). Note that the formulas for EC-EG and EC-P reflect the number of 163-bit and 2,048-bit modular multiplications, respectively.[5]

Parameters have been selected such as to obtain an equal 1,024-bit security level among all schemes and to reflect an envisioned WSN setting. For EC-NS, EC-OU, EC-P, and OU, primes p and q are selected such that $|n| = 1{,}024$, while we use one of the standard (IEEE) ECC curves over F_{163} defined in [11]. Random nonces are assumed to be 80 bits while plaintexts m are 8-bit values.

The results show that EC-EG benefits from its smaller modulus operations in both ciphertext size and computation efforts. However, the table does not reflect the costs for the demapping function. In a scenario where thousands of nodes send values in a big domain, EC-EG requires significantly more computation power for the decryption than other schemes, unless an improved mapping/demapping function can be found. In small WSNs, EC-EG can be recommended, especially if the decryption is performed on a powerful base station.

Further, the table shows that OU is the best scheme if EC-EG cannot be applied, e.g., in very large networks. EC-P provides the fastest decryption, while encryption and required bandwidth are not acceptable for constrained devices.

6 Key Management

Various key predistribution (KPD) schemes for wireless multihop ad hoc networks have been proposed. Although different KPD proposals support varying keying models like pairwise keying, groupwise keying, or a single network-wide key, the majority of the KPD proposals are designed for securing pairwise unicast traffic [13]. More concretely, by e.g., applying the concept of key rings, they ensure with a reasonable high probability the establishment of a trust relationship over various intermediate nodes to allow a secured unicast multihop channel. Such KPDs are most valuable in MANETs. Only a few KPDs have been proposed that support the encryption of convergecast traffic with in-network processing. We describe some KPDs for CDA by following the classification introduced in Section 2.

5. As modular multiplications with 2,048-bit moduli are approximately four times more expensive than with 1,024-bit moduli, we convert the fourteen 2,048-bit modular multiplications in the decryption in EC-P to fifty-six 1,024-bit modular multiplications.

6.1 KPD for Groupwise Keying

Currently, only one KPD for groupwise keying is known, which supports encryption transformations fulfilling (1).

6.1.1 Topology Aware Groupwise Keying

The *Topology Aware Group Keying* (TAGK) [32] supports the usage of a symmetric privacy homomorphic encryption transformation for securing convergecast traffic with in-network processing. TAGK distributes keys per "routable" region. The scheme is extremely robust against exhausting nodes, and it provides a higher system security compared to single-hop-based encryption approaches. However, since TAGK is designed to support a symmetric privacy homomorphic encryption transformation that requires the same key for *all* the encrypting parties that originate convergecast traffic, in particular for WSN applications requesting highest system security, there is a strong need for conceptual enhancements.

Before nodes are spread out over a geographical region, the manufacturer preconfigures at each node the same pool of keys and their key IDs. The key pool is limited by the storage space of the destination platform. Next, the WSN is rolled out such that all nodes are randomly distributed over a region, placing nodes in approximately uniform positions. Each node stores the same key pool and its key IDs. Once the nodes are spread out over a region, they remain static and the bootstrapping phase starts. This phase includes the election of active nodes, e.g., the first run of the *adaptive self-configuring sensor networks topologies* (ASCENT) protocol [9], and the election of aggregator nodes, e.g., with the *low energy adaptive clustering hierarchy* (LEACH) protocol [17] and a simple "going down" routing protocol initialization. In addition, a subset of nodes with distance $i = 1$ to the sink node randomly chooses a key list $\{k_1, \ldots, k_r\} \in K^*$ and locally broadcasts the key identifiers to nodes within distance $i + 1$. As a probability function of the distance i and the maximum expected distance l to the sink node, receiving nodes either delete the whole key pool or randomly choose one $k \in \{k_1, \ldots, k_r\}$ and delete the remaining keys. Nodes that did not receive a message $\mathrm{ID}_{k_1} \| \ldots \| \mathrm{ID}_{k_r}$ during a particular time frame after the network's roll out delete their key pool. This ensures that unreachable nodes do not store sensitive data.

6.2 KPD for Unique Keying

A KPD that supports unique keying is required in case the CDA encryption transformation is a PH from (2). Since

pairwise keys are used, the highest achievable system security is provided. Available candidates are given in Sections 6.2.1 and 6.2.2.

6.2.1 Random Unique Keying

In this KPD, keys are *randomly distributed* [7] to the nodes and only the sink node needs to store all the keys. Since the storage of keys on the nodes is independent of the final position of the nodes, the KPD reduces to a simple storage of different unique keys before the nodes' deployment. Obviously, such a simple KPD is nearly perfect for highly self-organizing distributed environments. In addition, from the security perspective, a pairwise keying model for convergecast traffic is preferable since it provides a higher system security. However, the benefit of the overall system security comes at the cost of additional overhead. First, the nodes' configuration before node deployment requires a pairwise pairing between each sensor node and the sink node to agree on the shared key. Second, since we are aiming at security solutions over a highly unreliable medium, one cannot ignore the impact of packet loss on the wireless broadcast medium. Revealing per data transmission the key IDs, respectively, node IDs of all the currently involved nodes therefore becomes mandatory. The usage of the CaMyTs scheme in the AIE operation mode is aiming at reducing the data overhead at the cost of reduced security. The required KPD for such a key setting can be achieved, e.g., by running the key management scheme from Eshenauer and Gligor [13].

6.2.2 Unique Keying with Algebraic Structuring

The *Topology Aware Unique Keying* (TAUK) [4] solution does neither require additional data overhead for key IDs when sending encrypted convergecast traffic nor does it require an extensive key setting before the node deployment. It can be used for a *double homomorphic encryption transformation* (DHET) [4]. One derivate of DHET is derived from the CaMyTs approach. At the same time, this approach is robust against exhausted nodes and an unreliable broadcast medium where sometimes data from a child node may not reach an aggregator node.

In the *initialization phase*, it is assumed that each sensor node N already knows its direct neighbors $Pred(N)$ and $Succ(N)$. Subsequently, each node receives a single symmetric key, whereas all keys from the sensor nodes are derived from a master key, which is solely stored at the sink node. In addition to its key, each node stores encrypted default values. Each of such ciphers corresponds to an $N' \in Succ(N)$. They provide robustness during the aggregation phase. During the initialization phase, the system is highly vulnerable, even to passive attacks. During an *aggregation phase*, convergecast traffic is encrypted end-to-end from the sensing nodes to the sink node. Each node N applies a PH by encrypting its monitored value with its own unique key and by subsequently summing up the resulting ciphertext to the received ciphertexts from its children $Succ(N)$. Since each node purely stores its own key, it cannot decrypt the incoming ciphers from its children. Only the sink node is enabled to decrypt the final aggregated value by applying the master key to the received ciphers. Since not always all nodes may have

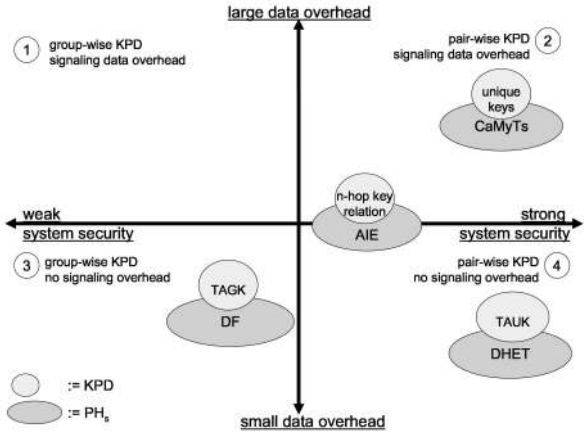

Fig. 3. Taxonomy of symmetric privacy homomorphic encryption schemes and their relation to KPD schemes.

contributed or due to interference on the wireless transmission medium packets may get lost, each intermediate node adds those stored default ciphers to the aggregated ciphered sum, which correspond to its direct children and which have not provided their input. During an aggregation phase, the system is secure against passive and active attacks.

6.3 Public Private Keying

In cases where an asymmetric additively homomorphic encryption transformation [see (3)] shall be applied to secure convergecast traffic within the WSN, it is preferable that a public/private key pair is generated at the sink node and the public key is loaded on each sensor node. Typically, this happens before the rollout of the sensor nodes. However, even a flooding of the public key after the deployment of the WSN is possible.

6.4 Classification

Fig. 3 classifies the discussed KPDs and the corresponding encryption transformations with respect to their provided system security and its data overhead. Since a key management based on public private keys does not fit to the criteria depicted in this figure, we do not consider it here. Note that a detailed discussion on the provided security of the concrete encryption transformation follows in Section 8. The CaMyTs [7] together with KPD unique keys provides strong security at the cost of high data overhead. It therefore belongs to category 2. Running it in the modes "n-hop key relation" with AIE [8] and OeMo [23] reduces the data overhead while it weakens the system security at the same time. We therefore see it at the edge between category 2 and category 4. The approaches [7] and [23] provide a moderate system security at a moderate to high data overhead during an aggregation phase. Whereas for PH [7] the KPD is as simple as possible since keys can be randomly stored on the node before deployment, OeMo [23] and AIE [8] require a complex and structured storing policy of keys without providing a clear solution how keys can be distributed in such a way.

7 Discussion of CDA Approaches

7.1 Evaluation of Security Aspects

In the following, we pick up the brief classification of attack scenarios described in Section 3 and evaluate to what extent the PH schemes are resistant or vulnerable. Additionally to the symmetric approaches DF, CaMyTs, and DF/CaMyTs, we evaluate EC-EG and OU as the most preferable asymmetric schemes.

7.1.1 Ciphertext Analysis

The analysis of eavesdropped ciphertexts is the most straightforward attack on cryptographic schemes. This is why it is absolutely necessary that the schemes are resistant to these attacks. To our knowledge, all discussed schemes are not vulnerable to this attack.

All five evaluated PH schemes, DF, CaMyTs, DF/CaMyTs, OU, and EC-EG, are probabilistic schemes, i.e., if the same plaintext is encrypted more than once it results in randomly distributed ciphertexts. This feature makes a cryptoanalysis much more complicated because statistical information is covered by the injected randomness. However, the source of the randomness seems to be the biggest threat concerning the pure ciphertext analysis. If it is possible to predict the next random numbers, it would imply a total break of OU, EC-EG, and CaMyTs and significantly weaken DF. Nevertheless, we consider all schemes as secure regarding the pure ciphertext analysis threat.

7.1.2 Known Plaintext Attack

Since in a WSN there are several ways of obtaining plaintext for a ciphertext, the known plaintext attacks are not only a hypothetical kind of attack.

Studies [31] show that in particular DF is very vulnerable to known plaintext attacks. Depending on the applied system parameters, a set of plaintexts with corresponding ciphertexts is sufficient to deduct the secret key. Wagner [31] also showed secure combinations of parameters. However, these parameters would imply message sizes of more than 1 Mbyte per message, which make them useless in WSNs. For CaMyTs, no actual attack of this type is known. However, with known plaintexts, an adversary can isolate the random streams, which could help deduct the key of the pseudorandom number generator.

With regard to DF/CaMyTs, we presume that there is no threat connected with known plaintexts. The DF part of the algorithm covers the random stream and the randomization of CaMyTs avoids the dangerous known plaintext vulnerability of DF.

Both asymmetric schemes (EC-EG and OU) are well-known and well-evaluated cryptographic algorithms. Since no known plaintext weaknesses have been published, it can be presumed they are secure regarding this attack. Anyway, the public key character of these schemes necessitates that the scheme is not vulnerable to known plaintext attacks because everyone is able to generate ciphertext/plaintext pairs.

7.1.3 Replay Attacks

Resending of previously sent packets with malicious intention is the easiest active attack. Consider the potential threats such replay attacks are posing (e.g., pretending in motion detection scenario), it is very desirable to have a protection against this issue.

CaMyTs and DF/CaMyTs have a resistance to such attacks, because every new ciphertext applies a new key. A previously sent packet encrypted with the current key will usually result in an unreasonable decrypted text, which can be recognized by the receiver.

DF, OU, and EC-EG do not have any inner protection against replayed or doubled packets. The additional integration of timestamps, sensor IDs, or a challenge/response system may help cope with the problem. However, the integration of such a feature or protocol in a CDA environment is not straightforward.

7.1.4 Malleability

In the description of this attack, we mentioned the example where the adversary wants to increase the measured temperature by 20 °C.

In EC-EG, such an attack can be performed very well. Consider the public key $(E, p, G, Y = xG)$ and the mapping function $M = mG$. Then, the encrypted message is the pair

$$[kG, kY + mG].$$

In order to obtain an encrypted value that is increased by 20, one can simply perform

$$[kG, kY + (m + 20)G] = [kG, kY + mG] + [0, 20G].$$

CaMyTs is also vulnerable to this attack. A ciphertext $(m + K_n) \bmod M$ can easily altered by

$$((m + 20) + K_n) \bmod M = (m + K_n) + 20 \bmod M.$$

For OU, the content m of the ciphertext $c = g^m h^r (\bmod\ n)$ can be modified by multiplying or dividing g that is part of the public key. Thus, OU is not secure against this attack.

DF and DF/CaMyTs are not vulnerable to this attack due to the architecture of the algorithm. In order to modify the content of a ciphertext, an adversary needs a part of the secret key (r).

7.1.5 Unauthorized Aggregation

This kind of attack that is very specific for PH schemes actually is only a variation of the malleability we described in Section 7.1.4. Instead of adding just 20, the idea of this attack is to add another ciphertext that contains 20. EC-EG allows one to add two messages unnoticed by doing

$$[(k_1 + k_2)G, (k_1 + k_2)Y + (m_1 + 20)G] = [k_1G, k_1Y + m_1G] + [k_2G, k_2Y + 20G].$$

However, the aggregated random parts $(k_1 + k_2)$ could somehow be noticed by the receiver, so that the interference may be detectable.

Likewise, OU and DF do not have any protection against unauthorized aggregation. An attacker can take any two ciphertexts and aggregate them without leaving marks.

Since the decryption of CaMyTs and DF/CaMyTs expects a specific embedded key, unauthorized aggregation would lead to damaged packets that do not contain a reasonable plaintext. This is why CaMyTs-based algorithms are not affected by this attack.

7.1.6 Forge Packets

Indeed, there is no need to modify existing packets if it is possible simply to generate proper ciphertexts. Naturally, asymmetric schemes like EC-EG and OU do not have any protection against the problem of forged packets. This is why in environments where the integrity of received messages is important EC-EG and OU must not be applied without additional protection. DF has secret parameters that are required for the encryption of a plaintext. Though extensive analysis shows that no approach has been published that allows one to generate proper ciphertexts without knowing these secret parameters. However, since every node in the system uses the same secret keys, it is considerable to use one (maybe captured) node as oracle that generates desired ciphertexts, even without having the keys.

Due to the fact that CaMyTs and DF/CaMyTs apply a new key for every message, these algorithms are resistant to the forged packet issue. Since additionally every node has an own stream of keys, it is not even possible to deduct useful information regarding one node from another one.

7.1.7 Physical Attacks

The group of physical attacks comprises all attacks on the actual node hardware in order to execute or support an attack. As already described, DF is severely vulnerable to such attack because the same secret system keys are used in every node. A revealed memory content can contain the secret key and thereby imply a total break of the system. Alternatively, captured or compromised nodes can be used as an encrypting or decrypting device.

To EC-EG and OU, such attack is not an important threat, unless the decrypting node (in most settings the sink node) that contains the private is attacked. Due to the asymmetric public key approach, an adversary cannot gain any additional information that can be used for further attacks.

Since CaMyTs does not use the same key on two encrypting nodes, a compromised node does not pose any additional risk to the system. However, one malicious node that injects bad messages may be a problem for the system. Though it is probably detected that something is wrong with the received message, without additional protocol it is neither possible to isolate the source of the malicious data nor to deduct the correct and usable ciphertext. To the best of our knowledge, it is an unsolved issue for all discussed PH schemes. DF/CaMyTs comprises the description of DF and CaMyTs. The DF part can be neglected if the memory is read, while the CaMyTs part of the algorithm is very secure.

7.2 Comparisons

Table 3 shows a brief evaluation of the described CDA encryption transformations regarding the set of properties and the described attack scenarios. Indeed, such an overview cannot deliver a satisfying assessment for every situation and parameter combination. For example, the ciphertext size of CaMyTs is considered as positive. However, the positive assessment is not justified anymore in case where many not responding IDs must be transmitted.

TABLE 3
Comparison of Various PH Algorithms

	DF	CaMyTs	ECEG	OU	CaMyTs/DF
Ciphertext size	−	+	o	−−	−
Effort encryption	o	+	−	−−	o
Effort decryption	o	−	−−	−−	−
Effort aggregation	o	++	−	−	o
Resistance to					
Ciphertext only	++	++	++	++	++
Chosen plaintext	−	+	++	++	++
Replay	−−	++	−−	−−	++
Malleability	+	−−	−−	−−	+
Mal. Aggregation	−	+	−−	−−	+
Forged packets	++	+	−−	−−	++
Captured sensors	−−	+	++	++	+

Another controversial point is the computation effort for EC-EG. Because ECC software implementations are known to be quite slow, it is assessed with "−". However, executed on hardware accelerators, ECC is very fast. Moreover, in this case, the power consumed during the computation is even smaller than it is required for the transmission of the encrypted data packet. Thus, if hardware accelerators are applied, the computation costs for ECC can be neglected [27].

Nevertheless, as a result of our evaluation, CaMyTs as the PH approach with the least computation efforts is also the most secure stand-alone PH approach. Its only real weakness is the malleability. In combination with DF as hybrid CaMyTs/DF even this weakness is solved. For the many benefits in the security category of the evaluation table, CaMyTs/DF has to pay in the efficiency category. The message size is bigger and the computation efforts are higher.

Actually, in many application scenarios not all properties must be perfectly fulfilled. In case only a simple encryption is wanted and an active attack, which is connected with considerable expenses, is not a probable threat, all four algorithms are reasonable. In such a case, side constraints could favor one algorithm or another. For example, since ECC is already part of the tinyOS operating system module tinySec, this makes the usage of EC-EG very reasonable.

More specific recommendations about possible application scenarios are presented in Section 7.3.

7.3 Application Recommendations

In a *synchronous sensor network*, the values are fluctual and adding security should not impact the reactive and real-time responsiveness of the system. For this reason, and due to the very restricted lifetime of the values, the authors support the application of symmetric PHs or a hybrid one with all its security weaknesses but with performance benefits. The matter of applying a similar scheme to *layered* topologies that require in-network decryption is one that we believe is not solvable by these approaches.

As to *asynchronous sensor networks*, we believe we have to consider the problem when applied to a *flat* and to a *layered* topology separately. *Layered* topologies may require

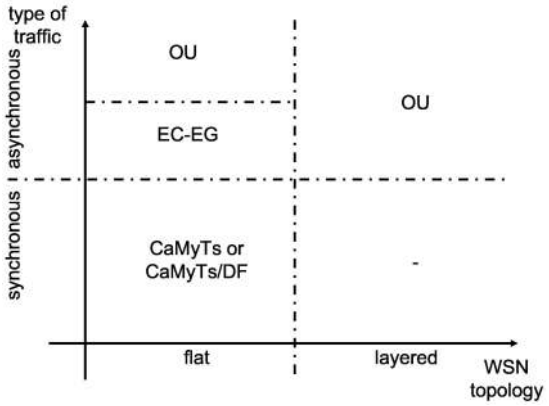

Fig. 4. Recommendations on the usage of additive homomorphic schemes in WSNs.

intermediate nodes to be able to decrypt values. In the previous sections, limitations on decryption, related to the value space and algorithm, are presented for EC-EG, which make the decryption operation unsuitable for implementation on the sensor nodes. We consider that in this case, the best candidate is OU since it provides the best ratio between encryption and decryption costs. This comes at the cost of a bigger ciphertext size, which we still consider acceptable for applications that require only seldom polling and aggregation of the values.

In a *flat* distribution of the network, we have to consider the threshold at which the constant addition of the values, or the initial size of the sensed values, affects the feasibility of decryption. The primary candidate, both in computation effort and bandwidth, is EC-EG. However, this scheme suffers from an expensive mapping function during decryption, which, in some cases, may become too costly to revert. Once this threshold is reached, we believe that OU is another possible candidate. The ciphertext size pushes it to a second, still viable, solution, where applying EC-EG is no longer possible. Fig. 4 provides a summary.

Finally, the application has a direct impact on the concrete CDA scheme to use. Calculating the minimum and maximum, as stated in [16], is not possible to achieve due to an inherent problem of using PHs. For such applications, we propose the usage of a scheme proposed in [1], which makes use of an Order Preserving Encryption Scheme [3] and applies it to a sensor network scenario. When calculating the variance, the problem of the value space appears once again: Since the aggregated data is actually the square of the sensed value, the value space doubles and may easily reach values no longer feasible for applying EC-EG. This case would be another possibility where OU can be applied.

8 Conclusion

CDA is a powerful mean for protecting WSNs with in-network processing. In this paper, we have discussed symmetric and asymmetric privacy homomorphic encryption transformations and gave recommendations regarding their usage in concrete application settings. Some are more preferable for usage in real-time responsive scenarios, whereas others are preferable in a time uncritical setting with a relatively seldom transmission of data. We have discussed a set of key management approaches that is particularly suited for CDA in WSNs.

Acknowledgments

The work presented in this paper was supported by the European Commission within the STReP UbiSec&Sens of the EU Framework Program 6 for Research and Development (IST-2004-2.4.3). 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 UbiSec&Sens project (http://www.ist-ubisecsens.org) or the European Commission. The authors would like to thank Joao Girao, Einar Mykletun, and Mithun Acharya who were involved in the earlier work, which provided the ground for this survey.

References

[1] M. Acharya, J. Girao, and D. Westhoff, "Secure Comparison of Encrypted Data in Wireless Sensor Networks," *Proc. Third Int'l Symp. Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2005.

[2] J.M. Adler, W. Dai, R.L. Green, and C.A. Neff, "Computational Details of the VoteHere Homomorphic Election System," *Proc. Ann. Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2000.

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order Preserving Encryption for Numeric Data," *Proc. ACM SIGMOD*, 2004.

[4] F. Armknecht, J. Girao, and D. Westhoff, "Topology Aware Key Management for Homomorphic Encryption of Convergecast Traffic in Wireless Sensor Networks," *Computer Comm.*, special issue on algorithmic and theoretical aspects of wireless ad hoc and sensor networks, 2008.

[5] J. Benaloh, "Dense Probabilistic Encryption," *Proc. Workshop Selected Areas of Cryptography (SAC '94)*, pp. 120-128, 1994.

[6] E.F. Brickell and Y. Yacobi, "On Privacy Homomorphisms," *Proc. Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '88)*, vol. 304, pp. 117-125, 1988.

[7] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient Aggregation of Encrypted Data in Wireless Sensor Networks," *Proc. Second Ann. Int'l Conf. Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous '05)*, July 2005.

[8] C. Castelluccia, *Cryptology ePrint Archive*, AIE, Report 2006/416, http://eprint.iacr.org/, 2006.

[9] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies," *IEEE Trans. Mobile Computing*, vol. 3, no. 3, pp. 272-285, July-Sept. 2004.

[10] D. Dolev and A.C. Yao, "On the Security of Public-Key Protocols," *IEEE Trans. Information Theory*, vol. 29, no. 2, pp. 198-208, 1983.

[11] J. Domingo-Ferrer, "A Provably Secure Additive and Multiplicative Privacy Homomorphism," *Proc. Fifth Information Security Conf. (ISC '02)*, pp. 471-483, 2002.

[12] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *Proc. Ann. Int'l Cryptology Conf. (CRYPTO '85)*, vol. IT-31, no. 4, pp. 469-472, July 1985.

[13] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. ACM Conf. Computer and Comm. Security (CCS '02)*, Nov. 2002.

[14] S. Galbraith, "Elliptic Curve Paillier Schemes," *J. Cryptology*, vol. 15, pp. 129-138, 2002.

[15] J. Girao, D. Westhoff, E. Mykletun, and T. Araki, "TinyPEDS: Tiny Persistent Encrypted Data Storage in Asynchronous Wireless Sensor Networks," *Elsevier Ad Hoc J.*, vol. 5, no. 7, pp. 1073-1089, Sept. 2007.

[16] J. Girao, D. Westhoff, and M. Schneider, "CDA: Concealed Data Aggregation for Reverse Multicast Traffic in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Comm. (ICC '05)*, May 2005.

[17] W.B. Heinzelmann, A.P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Trans. Wireless Comm.,* vol. 1, no. 4, pp. 660-670, Oct. 2002.
[18] J. Girao, D. Westhoff, and M. Schneider, "CDA: Concealed Data Aggregation in Wireless Sensor Networks," *Proc. ACM Workshop Wireless Security (WiSe '04),* poster, in conjunction with ACM MobiCom '04, Oct. 2004.
[19] K. Koyama, U.M. Maurer, T. Okamoto, and S.A. Vanstone, "New Public-Key Schemes Based on Elliptic Curves over the Ring Z_n," *Proc. Ann. Int'l Cryptology Conf. (CRYPTO '91),* pp. 252-266, 1991.
[20] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1996.
[21] E. Mykletun, J. Girao, and D. Westhoff, "Public Key Based Cryptoschemes for Data Concealment in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Comm. (ICC),* 2006.
[22] D. Naccache and J. Stern, "A New Public Key Cryptosystem Based on Higher Residues," *Proc. ACM Conf. Computer and Comm. Security (CCS '98),* pp. 59-66, 1998.
[23] M. Oenen and R. Molva, "Secure Data Aggregation with Multiple Encryption," *Proc. European Workshop Wireless Sensor Networks (EWSN '07),* Jan. 2007.
[24] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," *Proc. Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '99),* pp. 223-238, 1999.
[25] P. Paillier, "Trapdooring Discrete Logarithms on Elliptic Curves over Rings," *Proc. Ann. Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT '00),* pp. 573-584, 2000.
[26] S. Peter, P. Langendörfer, and K. Piotrowski, "On Concealed Data Aggregation for Wireless Sensor Networks," *Proc. Fourth IEEE Consumer Comm. and Networking Conf. (CCNC),* 2007.
[27] S. Peter, P. Langendörfer, and K. Piotrowski, "Public Key Cryptography Empowered Smart Dust Is Affordable," *Int'l J. Sensor Networks,* special issue on energy-efficient algorithm and protocol design in sensor networks, vol. 4, no. 1/2, 2008.
[28] R.L. Rivest, L. Adleman, and M.L. Dertouzous, "On Data Banks and Privacy Homomorphisms," *Foundations of Secure Computation.* Academic Press, pp. 169-179, 1978.
[29] T. Okamoto and S. Uchiyama, "A New Public-Key Cryptosystem as Secure as Factoring," *Proc. Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '98),* pp. 308-318, 1998.
[30] R.L. Rivest, L. Adleman, and M.L. Dertouzos, "On Data Banks and Privacy Homomorphisms," *Foundations of Secure Computation,* pp. 169-179, Academia Press, 1978.
[31] C. Wagner, "Cryptoanalysis of an Algebraic Privacy Homomorphism," *Proc. Sixth Information Security Conf. (ISC '03),* Oct. 2003.
[32] D. Westhoff, J. Girao, and M. Acharya, "Concealed Data Aggregation for Reverse Multicast Traffic in Wireless Sensor Networks: Encryption, Key Pre-Distribution and Routing," *IEEE Trans. Mobile Computing,* vol. 5, no. 10, pp. 1417-1431, Oct. 2006.

Steffen Peter received the diploma in computer science from the Brandenburg University of Technology (BTU), Cottbus, Germany, in 2006. After some preliminary work as student, he joined IHP, Frankfurt (Oder), Germany, in 2006. He worked in the Wireless Internet Project, developing a hardware TCP accelerator. In his diploma thesis, he was involved in the development of hardware cryptography accelerators. In this area, he has filed three patents and has authored two technical papers. He is currently a member of the mobile middleware group, where he is working in the research of solutions for security issues of wireless sensor networks. His research interests include security and privacy in mobile environments with emphasis on efficient hardware implementation for this purpose.

Dirk Westhoff received the PhD degree in computer science from the Distance University of Hagen in 2000. In 2007, he received a postdoctoral lecture qualification entitled "Security and Dependability Solutions for 4G Wireless Access Networks" from the Distance University of Hagen. Since 2001, he has been with the R&D Network Laboratories, NEC Europe, Heidelberg, Germany, where he is currently a chief researcher. Recently, he has been strongly involved in the definition and launching phases of the European projects UbiSec&Sens, SENSEI, and WSAN4CIP. He is cofounder of the European Workshop on Security in Ad Hoc and Sensor Networks (ESAS) series published by Springer. He has more than 50 peer-reviewed publications in network security and distributed system's security and is the holder of six patents. He has been involved in the TPC of several ACM and IEEE workshops and conferences. He is a member of the steering committee of the ACM Conference on Wireless Network Security (WiSec). His research interests include wireless security, ad hoc and sensor network security, and many other security and privacy aspects of distributed mobile communication. He is a member of the IEEE.

Claude Castelluccia received the engineering diploma in computer science from the Université de Technologie de Compiègne (UTC), Compiègne, France, the MSc degree in electrical engineering from the Florida Atlantic University, Boca Raton, Florida, in 1992, and the PhD degree from the Institut National de Recherche en Informatique et en Automatique (INRIA) in 1996. He was a postdoctoral researcher in the wireless research group of Stanford University in 1997. He has been a researcher at INRIA since 1997, where he is leading a research group on network security. He was a senior researcher at the University of California, Irvine from 2003 to 2005. He was appointed as an INRIA senior researcher (directeur de recherche) in 2005. He has published more than 100 scientific papers and several Internet drafts and RFCs. He also served on the technical program committee of several conferences and workshops in this field. He was a steering committee member of the European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS) and is currently a steering committee member of ACM Conference on Wireless Security (WiSec).