Securing Very Dynamic Groups and Data Aggregation in Wireless Sensor Networks

Claude Castelluccia
INRIA - 655 avenue de l'Europe 38334
Saint Ismier Cedex, France
Claude.Castelluccia@inria.fr

***Abstract*—**
We present a new encryption mode of operation that allows nodes of a network to exchange messages securely (i.e. encrypted and authenticated) without sharing a common key or using public key cryptography.

Our scheme is well adapted to networks, such as ad hoc, overlay or sensor networks, where nodes have limited capabilities and can share only a small number of symmetric keys. It provides privacy and integrity protection. We show that our proposal can be used in wireless sensor networks to send encrypted packets to very dynamic sets of nodes without having to establish and maintain group keys. These sets of nodes can be explicitly specified by the source or can be specified by the network according to some criteria, such as their location, proximity to an object, temperature range. As a result, a node can, for example, send encrypted data to all the nodes within a given geographical area, without having to identify the destination nodes in advance. Finally we show that our proposal can be used to implement a secure and scalable aggregation scheme for wireless sensor networks [1].

I. Introduction

Wireless sensor networks are often deployed in public and unattended environments. As a result, any malicious node can easily eavesdrop on communications and retrieve sensitive information. It is therefore essential to encrypt communications.

However encryption in wireless sensor networks is problematic since nodes have limited CPU and memory capabilities. Public key encryption is too CPU expensive and only symmetric cryptography algorithms can possibly be used. Symmetric cryptography requires the communicating nodes to share a common secret. Since wireless sensor networks are often very large and dynamic, it cannot be expected that each node be configured with a pairwise key with all the nodes of the network. Furthermore public key exchange protocols, such as Diffie-Hellman key exchange, require too much resource and are excluded. Some probabilistic key exchange protocols have been proposed [1], but they still require few protocol exchanges and are therefore not practical if the communication is short-lived (i.e. a node only wants to send one or two packets).

One common feature to many sensor networks is the need to communicate secretly with arbitrary sub-groups of sensors. Such communication may be short-lived and consist of only few messages. Furthermore the members of these groups might be very dynamic and unknown to the source. The group members might actually be defined on the basis of some criteria, such as location, proximity to an object, temperature range or any other environmental property. Existing group keying solutions [2] are not applicable to small and dynamic groups. Most of them assume that the group is rather stable, revocation is a rare event, and that the size of the group is quite close to the entire nodes population.

Our Contributions: We propose a new scheme that allows two nodes of a network to exchange messages securely (i.e. encrypted and authenticated) without sharing a common key or using public key cryptography. Our scheme also solves the short-lived group encryption problem described previously. It allows a node to send encrypted packets to very dynamic sets of nodes without having to establish or update group keys. These sets of nodes can be explicitly specified by the source or can be specified by the network according to some criteria, such as their location. As a result, a node can, for example, send encrypted data to all the nodes within a given geographical area, without having to identify the destination nodes in advance. Finally we show that our proposal can be used to securely aggregate encrypted data. The proposed scheme is based on stream ciphers and does not rely on any public key cryptography algorithms. It is therefore very efficient and well adapted to wireless sensor networks. It provides privacy and integrity protection.

Organization: The remainder of this paper is organized as follows. The section II presents our scheme, refered to as Authenticated Interleaved Encryption scheme. Section III shows how our proposal can be used to solve the short-lived secure group communication problem. Section IV proposes a novel secure and scalable aggregation scheme

[1]1-4244-1455-5/07/$25.00

for wireless sensor networks based on our new mode of operation. The related work is presented in Section V. Section VI concludes our work.

II. Authenticated Interleaved Encryption

This section describes the authenticated interleaved encryption mode of operation. A more detailed description of this scheme can be found in [3].

1) Terminology: The following notations appear in the rest of this paper.

- $k_{i,j}$: is the secret key shared between node N_i and node N_j.
- $kn_{i,j}$: is the secret key shared between node N_i and N_j's neighbors. This key is unknown to N_j.
- $iv_{i,j}$: is the Initialization Vector -IV- used by node N_i to encrypt a message for node N_j. Note that $iv_{i,j}$ must be unique for each message. It could, for example, be implemented as a counter.
- $Enc(k, m)$ is a stream cipher encryption algorithm that encrypts message m using the key k. For example the stream cipher RC4 can be used in our scheme. In this case, $Enc(k, m) = RC4(v, k) \oplus m$, where v is an initialization vector.
- $Dec(k, c)$ is the decryption algorithm that decrypts the cipher c into the plaintext m using the key k. If RC4 is used, $Dec(k, c) = c \oplus RC4(v, k)$.
- $C_{i,j} = Enc(k_{i,j}, m)$. $C_{i,j}$ is the result of the encryption of message m with the key $k_{i,j}$ and the IV v.
- $Cn_{i,j} = Enc(v, kn_{i,j}, m)$. $C_{i,j}$ is the result of the encryption of message m with the key $kn_{i,j}$ and the IV v.
- $C_{i,j}oC_{l,m} = Enc(v_i, k_{i,j}, Enc(v_l, k_{l,m}, m))$. Note that since we use a commutative steam cipher, $C_{i,j}oC_{l,m} = C_{l,m}oC_{i,j}$.

A. Assumptions/Security Model

1) Key distribution Model: We assume the *Leap-frog* key distribution model [4]. In this model, each node N_i shares a secret key, $k_{i,j}$ with each of its direct neighbors N_j. Furthermore node N_i shares a key, $kn_{i,j}$ with each of node N_j's direct neighbors. These keys are not known to node N_j.

These keys can be configured in a setup phase, when the nodes are first deployed, using the following three steps:

1) *Step1*: An 2-hop neighbor discovery protocol is run: N_i discovers its direct neighbors N_j, but also N_j's direct neighbors.
2) *Step2*: N_i then establishes pairwise keys with N_j and N_j's direct neighbors using, for example, a probabilistic key pre-distribution protocol [1].
3) *Step3*: N_i generates a random key $kn_{i,j}$ and unicasts it securely to each of N_j's neighbors by encrypting it with the pairwise keys that were established in the previous step.

This protocol is illustrated by Figure 1. Node N_1 discovers its direct neighbors (N_2 and N_3) and its 2-hop neighbors (N_5, N_6, N_7 and N_4) during step1. It then establishes pairwise key, $k_{1,i}$, with each of these nodes, i.e. with N_2, N_3, N_5, N_6, N_7 and N_4. It generates a random key, $kn_{1,2}$ and sends it to N_5 by encrypting it with $k_{1,5}$, to N_6 by encrypting it with $k_{1,6}$ and to N_4 by encrypting it with $k_{1,4}$.

We also assume that a node can check whether its neighbors have been revoked from the network. The details of the key establishment and revocation are out of the scope of this paper.

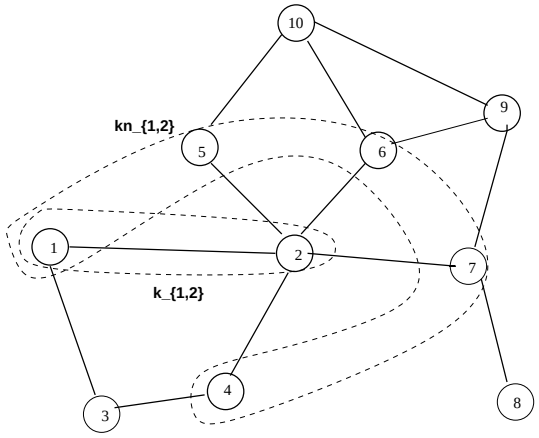

Fig. 1. **Key Distribution Model**. Node N_1 shares a secret key, $k_{1,2}$ with node N_2. It also shares a key, $kn_{1,2}$ with N_2's direct neighbors, i.e. N_5, N_6, N_7 and N_4. This key is not known to N_2.

2) Commutative Encryption: Our interleaved encryption proposal relies on a *commutative encryption* scheme i.e. an encryption scheme, denoted as $Enc(k, m)$, that satisfies the following property: $Enc(k1, Enc(k2, m)) = Enc(k2, Enc(k1, m))$.

Most block ciphers are not commutative. However stream ciphers are commutative. Stream cipher encryption and decryption operations are defined as follows:
$Enc(k, m) = m \oplus ks$ and $Dec(k, m) = m \oplus ks$, where ks is a keystream generated from a pseudo-random generator keyed with the secret key k. Since the xor (exclusive OR, $\oplus$) operation is commutative, stream ciphers are commutative. More specifically,:
$Enc(k1, Enc(k2, m)) = (m \oplus ks_2) \oplus k_1 = (m \oplus ks_1) \oplus ks_2 = Enc(k2, Enc(k1, m))$.

Some public key encryption schemes, such as the Pohlig Hellman encryption scheme, are also commutative, and could also be used in our scheme. However, we assume th use of a stream cipher in this paper.

B. Protocol Description

It is assumed, in this description, that a node, N_0, wants to privately send a message m to node N_D. All the nodes along the path from N_0 to N_D are denoted N_i, where N_0 is the source, N_1 the first node on the path,..., and N_D the final destination. Furthermore, each node on the path *only*

knows the next hop to the destination node N_D. We also assume that, as described in section II-A1, that each node N_i shares a pairwise key, $k_{i,j}$, with each of its neighbors N_j, and a private key, $kn_{i,j}$, with N_j's neighbors.

The protocol executed by each node N_i, on the path from N_0 to N_D, is described as follows:

if $(i == 0)$ **then**
  compute $C_0 = Enc(k_{s,1}, Enc(kn_{s,1}, m))$
  forward C_0 to N_1.
end if
if $1 \leq i \leq (D-1)$ **then**
  compute $t_i = Dec(kn_{i-2,i-1}, Dec(k_{i-1,i}, c_{i-1}))$
  compute $C_i = Enc(k_{i,i+1}, Enc(kn_{i,i+1}, t_i))$
  forward C_i to N_{i+1}
end if
if $(i == D-1)$ **then**
  compute $t_i = Dec(kn_{i-2,i-1}, Dec(k_{i-1,i}, c_{i-1})$
  compute $C_i = Enc(k_{i,i+1}, t_i)$
  forward C_i to N_{i+1}
end if
if $(i == D)$ **then**
  compute $m' = Dec(kn_{D-2,D-1}, Dec(k_{D-1,D}, c_{D-1})$
end if

C. Security Analysis

1) Privacy protection analysis: This section analyzes the security of the privacy/confidentiality of the Interleaved encryption scheme in presence of passive and actives attackers.

a) Passive attacks: The encryption scheme described in the previous section is secure against passive attackers. A node N_i, on the path, can retrieve $Enc(kn_{i-1,i}, m)$. However since it does not know $kn_{i-1,i}$ and since $Enc(.)$ is semantically secure, it cannot retrieve m. Furthermore an eavesdropper, that is not on the path, see, the message m encrypted three times with three different keys and cannot retrieve it.

b) Active attacks: As explained above, the protocol is secure against isolated compromised nodes. As in [4], the message m can only be retrieved if two adjacent nodes, one of them being on the path, collude. In fact, an attacker that compromises nodes N_i and N_{i-1} obtains the keys $kn_{i-1,i}$, $k_{i-1,i}$, $k_{i,i+1}$ and $kn_{i,i+1}$. When it receives the cipher $c_{i-1} = Enc(kn_{i-1,i}, Enc(k_{i-1,i}, Enc(kn_{i-2,i-1}, m)))$ from node N_{i-2}, it can recover the plaintext m. It therefore protect against passive attackers and isolated compromised (or curious) nodes.

Note that an active attacker (a compromised node) can modify the destination field of the message.The message will then be delivered to the wrong destination and recovered. This attack is possible because an intermediate node cannot authenticate the message and verify its integrity. In particular it cannot verify whether the node that forwards it the packet did not modify it (its destination address for example). The solution is to authenticate/sign the destination address such that intermediate nodes can reject packets whose destination address has been modified. However since the message changes at each intermediate node, message authentication code (MAC) cannot be used. The proposed solution is to link the encryption key and the destination address, D. For example, instead of using the key $k_{i,j}$,the key $k'_{i,j} = hash(k_{i,j}||D||nonce_{i,j})$ could be used, where $nonce_{i,j}$ is a unique random value or a counter sent together with the encrypted message. As a result if node N_{i-1} changes the destination address D with D_m, Node N_i will receive a message C_{i-1} and will decrypt it using $k_{i-1,i}$ and $kn_{i-2,i-1}$. However while N_{i-1} computed $kn_{i-2,i-1}$ using D, N_i will compute $kn_{i-2,i-1}$ using Dm. This decryption phase will then fail. As a result the message will be delivered to D_m, but D_m won't be able to correctly decrypt the message and retrieve the correct plaintext m.

2) Integrity Protection Analysis: This section analyzes the security of our proposal in term of integrity. The goal of an attacker is not, as in the previous section, to retrieve m but to modify m without the destination D noticing it.

Since message authentication is not provided, an active attacker can modify the message (and therefore the corresponding plaintext) in transit by modifying some bits. Since our scheme is based on a stream cipher, the destination has no way of detecting it. This is an attack on the integrity. This problem can be solved by having the source encrypting the message $R(m)||m$ instead of m, where $R(.)$ is a redundancy function. An attacker would then be able to modify m but not $R(m)$. Modification of the message could then be detected by the destination node [5]. It is believed that this combination of encryption and redundancy functions provide *plaintext integrity* (see remark 5.3 of [6]) if a stream cipher and a redundancy function such as AXU are used. Furthermore if the authentication tag is positioned *before* the message, instead of at the end, the AXU property is sufficient to provide cipher unforgeability (CUF-CPA). An encryption scheme is *ciphertext unforgeable* if it is infeasible for an attacker $\mathcal{F}$ that has access to an encryption oracle $\mathcal{O}_{\mathcal{ENC}}$ with the key k to produce a valid ciphertext under k not generated by $\mathcal{O}_{\mathcal{ENC}}$ as response to one of the queries by $\mathcal{F}$ [6]. Using the message length as an input of the redundancy function might also be a good design practice to avoid the attack described in [7].

III. Securing Very Dynamic Short-lived Multicast

A. Problem Statement

One important service common to many sensor networks is the need to privately communicate with arbitrary subsets of the network's sensors. These groups might actually be very dynamic and short-lived. In fact they might only consist of one command or one reply and, thus, a single message. Furthermore the members of group might be defined by the source or actually defined by the network on the basis of some arbitrary characteristic such as location, remaining battery power, and proximity to objects, temperature range or any other

environmental property (see example 2 described below). Existing group keying solutions [2] are not applicable to small and dynamic groups. Most of them assume that the group is rather stable, revocation is a rare event, and that the size of the group is quite close to the entire nodes population. These protocols require several rounds and are therefore not practical for short-lived groups. The only two possible approaches today are either to encrypt a message as many times as there are receivers (assuming the source shares a key with each of the receivers) or to enumerate all possible sensor subsets. Both solutions are clearly unpractical, highly inefficient and unworkable in any realistic sensor network. The first solution drastically increases the source load and the transmission bandwidth. The second one is highly non-scalable since all possible sensor subsets need to be pre-defined in advance. This is very difficult for large networks. Novel methods are definitely needed.

B. Protocol description

Our scheme can be used to design a solution to the short-lived multicast encryption problem. We assume that a random node S wants to send a message m securely to a subset, D, of nodes of the network. The nodes contained in this subset can be explicitly defined by the source, i.e. $D = \{N_1, N_2, ...N_j\}$ or can be implicitly defined by a criteria, i.e. "D=all the nodes that satisfy criteria C" (for example "C=all the nodes that are in the geographical area R"). In the later case, the decision is made locally by the forwarding nodes, as illustrated later by example 2. We also assume that the leap-frog key model, described in Section II-A1 is used, and that each node knows the next hop(s) toward the destination node(s).

The protocol is described as follows:

- S sends to each of its neighbors N_i a packet that is composed of a header that specifies D, the destinations nodes or the criteria, and the message m encrypted with the keys $k_{S,i}$ and $kn_{S,i}$ as described in Section II. Note that if N_i is part of the destination group, the message m is only encrypted once, with $k_{S,i}$.
- Upon reception of a packet, a node N_i verifies whether it is on the destination list (or if it satisfies the reception criteria i.e. it is within the area R). If this is the case, it decrypts the message once (with the key it shares with the forwarding node) and retrieves the message plaintext, m. It N_i is not on the destination list, it decrypts the message once or twice depending on whether it is only one-hop or several hops away from the source. It then encrypts twice for each of the neighbors that are on the multicast tree: with the keys it shares with its direct neighbor and two-hop neighbors. Note that if one of the neighbors is on the destination list (or satisfies the reception criteria), it only encrypts the message once (with the key it shares with this neighbor, as explained in the previous section).
- The message then propagates along the delivery tree until it reaches all destination nodes.

C. Examples

This section illustrates, via two examples, the proposed protocol. In the first example, the destination nodes are specified by the source. In the second example, the destination nodes are specified by the network, according to a criteria defined by the source.

1) Short-lived Multicast: Figure 2 illustrates how our scheme can be used to securely multicast a message m to a set of nodes. In this example, node N_1 sends a message to the list of nodes $D = \{N_8, N_9, N_{10}\}$.

N_1 encrypts m with $k_{1,2}$ and $kn_{1,2}$. It then sends the result, $C_{1,2}oCn_{1,2}$, to N_2 together with the list of destination nodes, D, to N_2. Upon reception of this message, N_2 identifies, using its routing protocol, the next node(s) toward the destination. In this example, the next node toward D is N_7. It then decrypts the message with $k_{1,2}$, and retrieves $Cn_{1,2}$ and encrypts the result with $k_{2,7}$ and $kn_{2,7}$. The resulting message, $c_2 = Cn_{1,2}oC_{2,7}oCn_{2,7}$, is forwarded to N_7. N_7 finds out that N_8 and N_9 are its direct neighbors and that N_{10} is reachable via N_9. It decrypts c_2 with $kn_{1,2}$ and $k_{2,7}$, and sends the result, $Cn_{2,7}$, encrypted with $k_{7,8}$ to N_8 and the result encrypted with $k_{8,9}$ to N_9. Since N_8 is on the destination list, it decrypts the message it received, i.e. $Cn_{2,7}oC_{7,8}$ with $k_{7,8}$ and $kn_{2,7}$ to retrieve m. Similarly, since N_9 is on the destination list, it decrypts the message it received, i.e. $Cn_{2,7}oC_{7,9}$, with $k_{7,9}$ and $kn_{2,7}$ to retrieve m. N_9 finally encrypts m with $k_{9,10}$ and forwards the result to N_{10} that can also retrieve m.

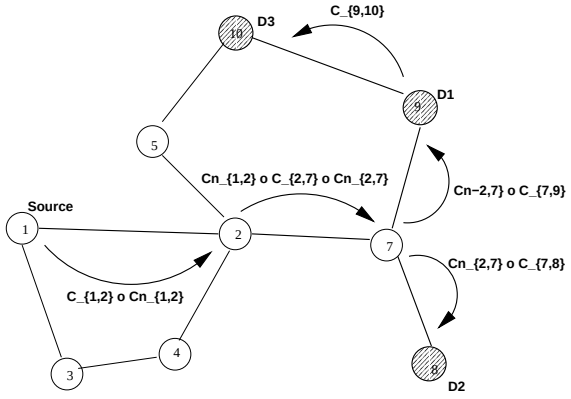

Fig. 2. **Short-lived Multicast**. Node N_1 multicasts a secret message to nodes N_8, N_9 and N_{10}.

2) Geographical Short-lived Multicast: Figure 3 illustrates the protocol when the source does not specify the list of destination nodes but instead a reception criteria. In this example, the reception criteria is "all the nodes within the geographical area R". Both nodes N_5 and N_{10} satisfy this criteria. We assume that each node is able to

verify whether its direct neighbors and itself satisfy the reception criteria.

The source, N_1, encrypts m with $k_{1,2}$ and $kn_{1,2}$. It then sends the result, $C_{1,2}oCn_{1,2}$ to N_2 together with the criteria, D, to N_2. It also sends the message m, encrypted with $k_{1,3}$ and $kn_{1,3}$, to N_3. However for simplicity sake, we only consider the messages that are on the path to the destination nodes. Upon reception of this message, N_2 finds out that its neighbor N_5 satisfies the criteria. It therefore decrypts the message with $k_{1,2}$ (and retrieve $Cn_{1,2}$) and encrypts the result with $k_{2,5}$. The resulting message, $c_2 = Cn_{1,2}oC_{2,5}$, is forwarded to N_5 together with D. N_2 also, possibly, encrypts $Cn_{1,2}$ with $k_{2,7}$ and $kn_{2,7}$ and forwards the results to N_7. Upon reception of the message, N_5 finds out that it satisfies the reception criteria. It therefore decrypts the message with $kn_{1,2}$ and $k_{2,5}$ to retrieve the message m. Since its neighbors D_{10} also satisfies the reception criteria, it forwards the message m encrypted with $k_{5,10}$ to it. D_{10} can then retrieve the message m.

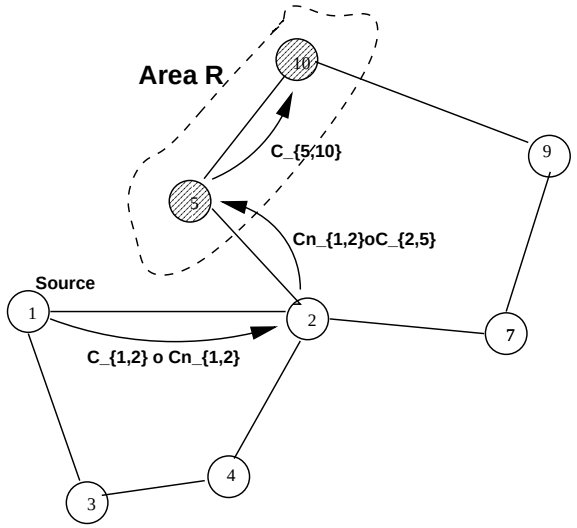

Fig. 3. **Geographical Short-lived Multicast**. Node N_1 multicasts a secret message to all nodes with area R, i.e. nodes N_5 and N_{10}

D. Security Analysis

Our scheme is secure against passive attackers and isolated active attackers. An attacker that compromises two adjacent nodes (or n adjacent nodes, according to the applications) on the path can recover the encrypted messages. However, an isolated node, even compromised, that is listening to communications is unable to decrypt them. As a result, the security provided by our solution is better than the security of schemes based on global group keys (all the network nodes share a common group key) or on hop-by-hop encryption (the messages are decrypted/encrypted hop-by-hop until they reach the destination). With these schemes, any curious or compromised node can listen to the communications. On the other hand, our scheme is less secure than schemes that use end-to-end encryption. However, as argued earlier in this paper, it is not always feasible and practical to establish secret keys between communicating nodes in constrained environments, such as wireless sensor networks, especially for short-lived communications. Furthermore, current secure group communication solutions are not adapted to very dynamic and short-lived groups since they assume that group members are stable and revocations are rare events. To our knowledge, our proposal is the first scheme to provide a solution to the challenging short-lived secure broadcast problem.

IV. Securing Data Aggregation in WSN

A. background and problem statement: the $Saggr$ scheme

Wireless sensor networks (WSNs) are composed of tiny devices with limited computation and energy capacities. For such devices, data transmission is a very energy-consuming operation. It thus becomes essential to the lifetime of a WSN to minimize the number of bits sent by each device. One well-known approach is to aggregate sensor data (e.g., by adding) along the path from sensors to the sink. Aggregation becomes especially challenging if end-to-end privacy between sensors and the sink is required, i.e. when the communication between each sensor and the sink is encrypted using a key that is not known to the aggregators.

[8] proposes a simple and provably secure additively homomorphic stream cipher that allows efficient aggregation of encrypted data. This new cipher, referred to as $Saggr$, only uses modular additions (with very small moduli) and is therefore very well suited for CPU-constrained devices.

The main idea of the scheme presented in [8] is to replace the xor (Exclusive-OR) operation typically found in stream ciphers with modular addition ($+$), as described by Table I.

Additively Homomorphic Encryption Scheme
Encryption:
1) Represent message m as integer $m \in [0, M-1]$ where M is large integer.
2) Let k be a randomly generated keystream, where $k \in [0, M-1]$
3) Compute $C = Enc(k, m) = m + k \pmod M$
Decryption:
1) $Dec(k, C) = C - k \pmod M$
Addition of Ciphertexts:
1) Let $C_1 = Enc(k_1, m_1)$ and $C_2 = Enc(k_2, m_2)$
2) For $k = k_1 + k_2$, $Dec(k, C_1 + C_2) = m_1 + m_2$

TABLE I

It is assumed that $0 \leq m < M$. Due to the commutative property of addition, the above scheme is additively homomorphic. In fact, if $C_1 = Enc(k_1, m_1)$ and $C_2 =$

$Enc(k_2, m_2)$ then $C_1 + C_2 = Enc(k_1 + k_2, m_1 + m_2)$. 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 results 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 identifier. This secret key pre-computed and shared between the node and the sink, while the message id can either be included in the query from the sink or derived from the time period in which the node is sending its values in (assuming some form of synchronization).

It is shown that aggregation based on this cipher can be used to efficiently compute statistical values such as mean, variance and standard deviation of sensed data, while achieving significant bandwidth gain. To compute the mean, each sensor encrypts its data x_i to obtain $C_i = Enc(k_i, x_i)$, k_i is the key it shares with the sink. It then forwards C_i to its parent, who aggregates all the C_j of its k children by simply adding them up (this addition is performed modulo M). The resulting value is then forwarded. The sink ends up with value $C = \sum_{i=1}^{n} C_i \pmod M$. It can then compute $S = Dec(K, C) = C - K \pmod M$, where $K = \sum_{i=1}^{n} k_i$, and derive the average as follows: Avg $= S/n$.

One limitation of this proposal is that the identities of the non-responding 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. We show in the following section that Interleaved Encryption can be used to solve this problem.

B. AIE: Aggregation of Interleaved Encrypted Data

This section presents the AIE scheme and explains how the $Saggr$ scheme can be used in an interleaved encryption mode to solve the identity transmission and scalability problems described in the previous section.

1) Assumptions: We assume for simplicity and without loss of generality that the network is structured as a tree. The sink is a top, the aggregators are the intermediate nodes and the sensors are the leaves. Sensors encrypt their sensed data and send the result to their local aggregator. Each aggregator securely aggregates the data it receives from its children and forwards the results to the next aggregator (i.e. its parent in the tree) toward the sink.

We also assume that each node shares a pairwise key with its direct parent, its 2-hop parent, 3-hop parent,...and n-hop parent, where n is a system parameter. Figure 4 illustrates this key model. In this example, n is set to 3. Each node shares a pairwise key with its parent, 2-hop parent (grand-parent) and 3-hop parent. For example, Node 1 shares a key with nodes 5, 7 and 9. These keys can be establishment using a scheme such as [1].

2) AIE protocol description: When a sensor, N_i, sends a message, its encrypts it n-time using the additively homomorphic scheme described in [8], i.e. $Saggr$. The first time with the key it shares with its direct parent, the second time with the key it shares with its 2-hop parent,..., the n^{th} 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, $2 - hop$ children,...,$n - hop$ children. The result is then encrypted n times with the keys it shares with its parent, 2-hop parent,..., and n^{th}. 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.

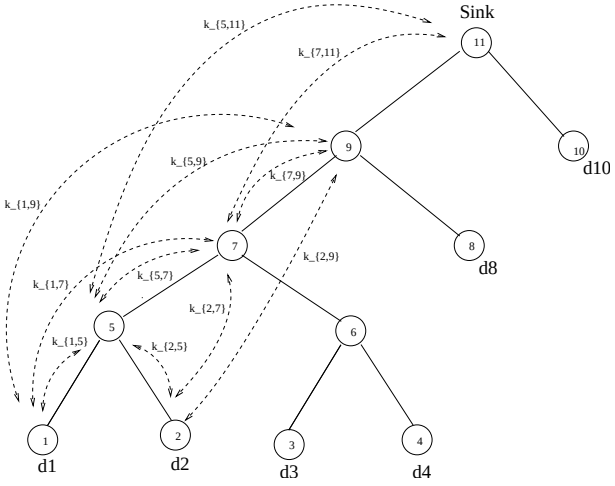

Fig. 4. **Aggregation of Interleaved Encrypted Data: Key model**.

3) Example: Let's consider the network defined on Figure 4 and let's assume that the sensors N_1, N_2, N_3, N_4, N_8 and N_{10} send their data, respectively d_1, d_2, d_3, d_4, d_8 and d_{10}, to the sink (N_{11}). The nodes N_5, N_6, N_7, N_9 are aggregators.

Node N_1 encrypts its data d_1 3 times: with $k_{1,5}$, $k_{1,7}$ and $k_{1,9}$, and sends the result c_1 to N_5 together with its identifier. Similarly N_2 encrypts its data d_2 with $k_{2,5}$, $k_{2,7}$ and $k_{2,9}$, and sends the result c_2 to N_5 together with its identifier. N_5 decrypts c_1 with $k_{1,5}$ and c_2 with $k_{2,5}$ and adds the results. It then encrypts the sum with $k_{5,7}$, $k_{5,9}$ and $k_{5,11}$ and sends the result, c_5 together with the identifiers of N_1 and N_2 to N_7. Similarly, node N_3 encrypts its data d_3 with $k_{3,6}$, $k_{3,7}$ and $k_{3,9}$, and sends the result c_3 to N_6 together with its identifier. N_4 encrypts its data d_4 with $k_{4,6}$, $k_{4,7}$ and $k_{4,9}$, and sends the result c_4 to N_6. N_6 decrypts c_3 with $k_{3,6}$

and c_4 with $k_{4,6}$ and adds the results. It then encrypts the sum with $k_{6,7}$, $k_{6,9}$ and $k_{6,11}$ and sends the result, c_6 together with the identifiers of N_2 and N_3 to N_7. As a result, N_7 receives c_5 from N_5 and c_6 from N_6. It decrypts c_5 with $k_{5,7}$, $k_{1,7}$ and $k_{2,7}$ and obtains the value x. It then decrypts c_6 with $k_{6,7}$, $k_{3,7}$ and $kn_{4,7}$ and obtains the value y. It encrypts the sum of x and y with $k_{7,9}$ and $k_{7,11}$, and sends the results c_7 to N_9 together with the identifiers of N_5, N_6, N_1, N_2, N_3 and N_4. N_8 encrypts its data d_8 with $k_{8,9}$, $k_{8,11}$ and sends the result c_8 to N_9. N_9 adds c_5 and c_8 and decrypts the results with $k_{7,9}$, $k_{8,9}$, $k_{6,9}$,$k_{5,9}$,$k_{4,9}$,$k_{3,9}$,$k_{2,9}$ and $k_{1,9}$. It then encrypts the results with $k_{9,11}$ and obtains c_9. c_9 is then sent to N_{11} together with the identifier of N_8, N_7, N_6 and N_5. Note that the identifiers of N_1, N_2, N_3 and N_4 do not need to be forwarded anymore. The sink N_{11} decrypts the message it received with $k_{9,11}$, $k_{8,11}$, $k_{7,11}$, $k_{6,11}$ and $k_{5,11}$ and retrieves the sum of the plaintext values i.e $d_1+d_2+d_3+d_4+d_5+d_6+d_7+d_8+d_9$.

4) Performance-Security tradeoff: With the previous 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 original scheme. In the original 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 has 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. With the example of Section IV-B3, each node has to forward at most 6 identities if AIE is used. With the original scheme, this number can go up to 12.

However the AIE based scheme is less secure than the $Saggr$ scheme. An attacker that corrupts n consecutive nodes can actually retrieve the aggregated value at the lowest corrupted aggregator in the tree. For example, if an attacker corrupts node 5, 7, and 9, it is able to retrieve the aggregated value available at node 5, i.e. d_1+d_2. With the original scheme, corrupting aggregators does not reveal any information about the aggregated value.

There is a clear tradeoff between the number of identities to be forwarded (i.e. bandwidth cost) and security. By decreasing n, the bandwidth cost decreases. However the security is also reduced because the number of consecutive nodes to corrupt is smaller. By increasing n, the bandwith cost and security increase. If $n = 1$, our scheme is similar to hop-by-hop encryption. This configuration is optimal in term of bandwidth but very weak security-wise. On the other hand, if $n = h$ (where h is the number of level in the tree), our scheme is similar to the original aggregation scheme of [8]. Its bandwidth cost is high but its security is maximum.

C. Hybrid AIE (HAIE)

A possible extension to the previous scheme is to combine the AIE protocol, with the secure aggregation scheme described in [8]. The main steps of this new protocol, referred to as $HAIE$ (Hybrid AIE), are described as follows:

1) Each node uses the AIE protocol to transmit their data to the sink.
2) A small percentage of nodes, the $Rnodes$, randomly chosen at each aggregation epoch, additionally encrypts their data with the key they share with the sink. These nodes encrypts their data $n + 1$ times.

As an example, let's consider the network described on Figure 5 and let's assume that $N2$ is the $Rnode$. Each node encrypts their data using AIE as explained above. $N2$ additionally encrypts its data with $k_{2,11}$, i.e. the key it shares with the sink: N_2 sends $d_2+k_{2,5}+k_{2,7}+k_{2,9}+k_{2,11}$ to N_5. It also needs to send its identifier id_2 along.

The message is then processed according to the AIE protocol until the sink. The sink decrypts it using the AIE protocol. In addition, it decrypts the result one more time using $k_{2,11}$ in order to recover the aggregated plaintext data. This scheme slightly increases the bandwidth cost, because id_2 has be appended to the message until the sink. However, it increases security drastically: even if the attacker corrupts 3 consecutive nodes (let's say N_1, N_5 and N_7), it won't be able to recover the data without corrupting N_2. Since the $Rnodes$ are constantly changing, the attack is much more difficult to perform.

In order to evaluate the security gain and the bandwidth cost of this optimization, we have performed several simulations. We considered a WSN composed of 2048 nodes. The WSN was structured as a tree rooted at the sink, and each node sent their data along the tree toward the root.We simulated the $Saggr$, the AIE (with n set to 2) and the $HAIE$ (with n set to 2) schemes. We then considered a strong attacker that always corrupts sets of n consecutive nodes, and then evaluated for each of the scheme their security and bandwidth cost. The security of a scheme is computed by counting the total number of compromised nodes, as a result of the corruption of x nodes. For example with the AIE scheme, if $n = 2$ and if N_7 and N_9 are corrupted (see Figure 5), then all the nodes rooted at N_7 are considered compromised (in addition to N_7 and N_9) because the attacker can get their aggregated plaintexts. With the $HAIE$ scheme, the nodes rooted at N_7 are considered compromised only if this subtree does not contain any non-corrupted $Rnode$. With the $Saggr$ scheme, the number of compromised nodes is equal to the number of corrupted ones.

The bandwidth cost is computed by adding the number of identifiers sent per link on the whole networks.

The simulation results are reported on tables II and III. Table II displays the number of compromised nodes with the $HAIE$ scheme for different numbers of $Rnodes$

when 90 nodes are corrupted. It also displays the corresponding bandwidth costs. It should be noted that with the $Saggr$ scheme the number of compromised nodes is only equal to 90, but the bandwidth cost is equal to 20267. With the AIE scheme, the number of compromised nodes is large and equal to 450, but the bandwidth cost is small and equal to 4050. With the $HAIE$, the bandwidth cost increases slightly (compared to AIE), but the security improves significantly: the number of compromised nodes ranges between 195 and 125, which is closer to what is achieved with the $Saggr$ scheme. The values between parentheses represent the security gain and bandwidth loss of the $HAIE$ scheme compared to the AIE scheme. For example, when 50 Rnodes are used, $HAIE$ reduces the number of compromised nodes by 56.6% (from 450 to 195), and increases the bandwidth cost by only 13.5% (from 4050 to 4547). The $HAIE$ scheme provides the best performance-security tradeoff: it improves security, without significantly increasing the bandwidth cost.

Table III shows similar results when 300 nodes are corrupted. With the $Saggr$ scheme the number of compromised nodes is then equal to 300, whereas with the AIE scheme, it is equal to 4050.

TABLE II
HAIE SIMULATION RESULTS (NB. CORRUPTED NODES= 90)

	nb. of Rnodes		
	50	100	300
Sec.	195 (-56.6%)	167 (-62.8%)	125(-72%)
Bw	4547 (+13.5%)	5042 (+25%)	7032 (+73%)

TABLE III
HAIE SIMULATION RESULTS (NB. CORRUPTED NODES= 306)

	nb. of Rnodes		
	50	100	300
Sec	671 (-50%)	567 (-58.6%)	422 (-68.7%)
Bw	4547 (+13.5%)	5042 (+25%)	7032 (+73%)

Table IV displays the bandwidth cost of each scheme at each level of the aggregation tree. It shows, as expected, that the AIE cost is constant and equal to 3. The bandwidth costs of the $Saggr$ and $HAIE$ schemes increase for nodes at higher level (i.e. closer to the sink) because they must forward the identifiers of the nodes in their subtree. The cost of the $HAIE$ scheme, while higher than the AIE scheme, is still reasonable.

D. Data Integrity Protection

The authenticated scheme proposed in Section II-C2 cannot be used with aggregation. In fact, since the plaintext is changing as the data are being aggregated (i.e. added) and the aggregators do not have access to the plaintext data, the redundancy check will fail. Furthermore, by definition, the redundancy function cannot be additively-homomorphic. Therefore only the basic scheme, as defined in Section II-B is applicable.

We suggest, in this case, to authenticate the messages hop-by-hop to prevent external attackers from modifying

TABLE IV
NUMBER OF BITS SENT PER NODE (# RNODE=200)

Levels	Num Nodes	Saggr	AIE	HAIE
1	1024	1	1	1.07
2	512	3	3	3.2
3	256	7	3	3.5
4	128	15	3	4.0
5	64	31	3	5.1
6	32	63	3	7.5
7	16	127	3	12.1
8	8	254.8	3	21.5
9	4	510.7	3	40.25
10	2	1022.5	3	77.5
11	1	2046	3	153

messages. This protects against external attackers but not compromised nodes. A compromised node can modify the encrypted message but still computes the correct authentication tag. The modification will then be undetected. However, we argue that this does not reduce security since when aggregation is performed any aggregator or sensor can add arbitrary value to the plaintext and falsify the aggregated value. Note that such an attack does actually not require the attacker to compromise any node since the sensed data can itself be modified. As explained in [9], other techniques are needed to verify the plausibility of the resulting aggregate and increases the aggregation resiliency. In WSNs, authentication does not provide data authenticity, but can instead be used to enforce access control, i.e. to prevent unauthorized nodes from injecting fake packets in the networks. This access control can efficiently be performed with hop-by-hop authentication and does not require end-to-end authentication. Hop-by-hop authentication is therefore sufficient and is probably the best level of integrity protection that can be provided with secure aggregation.

V. RELATED WORK

There have been several new key establishment proposals for wireless sensor networks recently. Most of them are based on the random key pre-distribution that was proposed by Eschenauer and Gligor [1]. In this scheme, each sensor is configured with a random subset of a large pool of keys. These keys are used for point-to-point security by having a sender use a key known to be shared by the receiver. Chan et al. [10] improves and analyzes this scheme. Du et al. [11] extends the Eschenauer-Gligor scheme with a Blom pairwise key-generation scheme. Liu and Ning [12] proposes a polynomial-based key distribution scheme. All of these schemes are pretty effective for point-to-point communications. However most of them require some message exchanges and are therefore not adapted to short-lived communications. Furthermore none of them are applicable to group communications.

Current group keying schemes either rely on some Diffie-Hellman group extensions [13], [14], and are therefore not adapted to sensor networks, or requires assume that the groups are pretty stable [2]. These later schemes,

referred to as "broadcast encryption", are not applicable when the receiver subset is much smaller than the entire sensor population.

Vogt [15] and Zha et al. [16] have proposed data integrity protection schemes for secure node-to-node communications based on interleaved authentication. Goodrich [4] extended this work to broadcast and group integrity. These schemes are very related to our proposal but do not provide privacy protection. AIE builds on these approaches to provide authenticated and encrypted group security.

Önen and Molva describe in [17] a secure aggregation scheme that is also based on the homomorphic scheme described in [8] and use multiple encryption. This proposal is very similar to the basic protocol described in Section IV-B and was developed in parallel and independently of our work. We propose an optimization, the $HAIE$ protocol, that significantly improves the security, at a marginal bandwidth cost.

VI. Conclusion

This paper presents a new scheme that allows CPU and storage constrained nodes of a network to securely exchange messages (i.e. encrypted and authenticated) without sharing a common key or using public key cryptography. Our scheme can, for example, be used by a sensor to send secret data to a node it does not share a security association with or that are defined by the network according to some criteria (location, functionality). It can also be used to solve the so-called "short-lived broadcast encryption" problem. It allows a node to send encrypted packets to very dynamic sets of nodes without having to establish or maintain group keys. These sets of nodes can be explicitly specified by the source or can be specified by the network according to some criteria, such as their location, proximity to an object or functionality (i.e. aggregators/sinks). As a result, a node can, for example, send encrypted data to all the nodes within a given geographical area, without having to identify the destination nodes in advance.

We also show that our proposal can be used to implement a secure and scalable aggregation scheme for wireless sensor networks. Our scheme improves the bandwidth performance of the secure aggregation scheme described in [8] and the security performance of the scheme described in [17]

Finally, note that Authentication Interleaved Encryption can optionally provide source anonymity, since the destination does not need to know the source to decrypt the message. This might be a useful feature for some applications.

Acknowledgments

I am thankful to Aurelien Francillon for his suggestions that helped improve the paper. I would also like to thank the anonymous reviewers for their comments and suggestions.The work described in this paper is based on results of IST FP6 STREP *UbiSec&Sens* (http://www.ist-ubisecsens.org). *UbiSec&Sens* receives research funding from the European Community's Sixth Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

[1] L. Eschenauer and V. D. Gligor, "A Key Management Scheme for Distributed Sensor Networks," *ACM CCS*, pp. 41–47, 2000.

[2] Sandro Rafaeli and David Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys*, vol. 35, pp. 309–329, 2003.

[3] Claude Castelluccia, "Authenticated interleaved encryption," Cryptology ePrint Archive, Report 2006/416, 2006, http://eprint.iacr.org/.

[4] M. Goodrich, "Leap-frog packet linking and diverse key distributions for improved integrity in network broadcasts," in *IEEE Security and Privacy*, May 2005.

[5] Jee An and Mihir Bellare, "Does encryption with redundancy provide authenticity?," in *EUROCRYPT*, 2001.

[6] Hugo Krawczyk, "The order of encryption and authentication for protecting communications," in *CRYPTO*, 2001.

[7] David Wagner, "Attacks on the hash-then-encrypt (for stream cipher)," http://www.cs.berkeley.edu/ daw/my-posts/mdc-broken2.

[8] Claude Castelluccia, Einar Mykletun, and Gene Tsudik, "Efficient aggregation of encryption data in wireless sensor networks," in *IEEE Mobiquitous*, 2005.

[9] David Wagner, "Resilient Aggregation in Sensor Networks," *Workshop on Security of Ad Hoc and Sensor Networks*, 2004.

[10] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Security and Privacy Symposium*, 2003.

[11] W. Du, J.Deng, Y. Han, and P. Varshney, "A pairwise predistribution scheme for wireless sensor networks," in *ACM Conf. Computer and Communication Security*, 2003.

[12] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *ACM Conf. Computer and Communication Security*, 2003.

[13] Michael Steiner, Gene Tsudik, and Michael Waidner, "Key agreement in dynamic peer groups," in *IEEE Transactions on Parallel and Distributed Systems*, July 2000.

[14] Yongdae Kim, Adrian Perring, and Gene Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *ACM CCS*, November 2000, pp. 235–244.

[15] Harald Vogt, "Integrity preservation for communication in sensor networks," Tech. Rep. 434, ETH Zurich, Institute for Pervasive Computing, Feb. 2004.

[16] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks," *Security and Privacy*, 2004.

[17] Suna Melek Önen and Refik Molva, "Secure data aggregation with multiple encryption," in *EWSN 2007, European Wireless Sensor Networks, January 29-31, 2007, Delft, The Netherlands*, 2007.