Authenticated Interleaved Encryption
and its Application to Wireless Sensor Networks

Claude Castelluccia, INRIA
claude.castelluccia@inria.fr

***Abstract*—**

We present AIE (Authenticated Interleaved Encryption), a new scheme 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. An eavesdropper listening to a communication is unable to decrypt it and modify it without being detected. 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.

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 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.

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. Novel methods are needed.

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.

Our 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. It is secure against passive eavesdroppers, and secure against active attackers as long as no more than n consecutive nodes get compromised, where n is a system parameter.

Organization: The remainder of this paper is organized as follows. Section III-B introduces our scheme in detail. It presents the *Interleaved Encryption* technique. It then introduces a variant, the *Authenticated Interleaved Encryp-*

tion scheme, that provides authentication in addition to privacy protection. Section III shows how the AIE proposal can be used to solve the short-lived secure group communication problem. It also explains how our proposal can be useful to implement a secure and scalable aggregation scheme for wireless sensor networks. The related work is presented in Section IV. Section V concludes our work.

II. Interleaved Encryption

A. Assumptions/Security Model

We describe the assumptions regarding the network before we present our schemes in detail.

1) Key distribution Model: We assume the *Leap-frog* key distribution model [3]. In this model, each node i shares a secret key, $k_{i,j}$ with each of its direct neighbors j. Furthermore node i shares a key, $kn_{i,j}$ with each of node j's direct neighbors. These keys are not known to node j (see Figure 1).

These keys can be configured in a setup phase, when the nodes are first deployed, by a network administrator or using some kind of probabilistic key pre-distribution protocols [1]. 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.

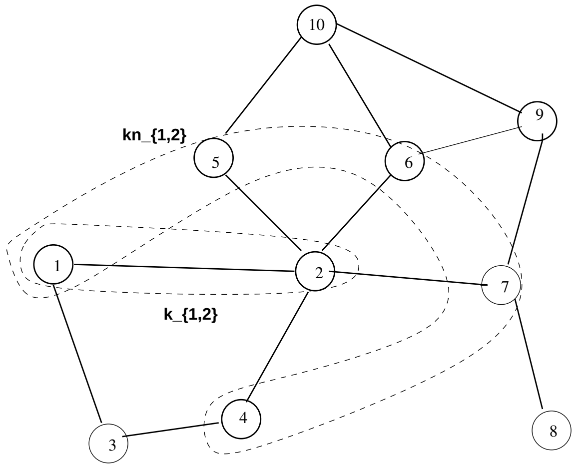

Fig. 1. **Key Distribution Model**. Node N_1 shares a secret key, $k_{1,2}$ with node N_2. It also shares a key, $kn_{1,2}$ with N_2's direct neighbors, i.e. N_5, N_6, N_7 and N_4. This key is not known to N_2.

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 stream key 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. For simplicity, we assume a stream cipher in the description of our scheme.

3) Security Model: Our goal is to design a scheme that allows a node S to send a message m privately to a node D without having to establish a secret key. Our scheme must be secure against passive and active attackers. A passive attacker should not be able to decrypt and recover the message. An active attacker should not be able to modify the message without being detected and should not be able to recover the message m.

4) 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.
- $Enc(v, k, m)$ is a stream cipher encryption algorithm that encrypts message m using the key k and the IV v. For example the stream cipher RC4 can be used in our scheme. In the case, $Enc_k(v, k, m)$ = *RC4(v,k) xor m*.
- $Dec(v, k, c)$ is the decryption algorithm that decrypts the cipher c into the plaintext m using the key k and the IV v. If RC4 is used, $Dec(v, k, c)$= *c xor RC4(v,k)*.
- $C_{i,j} = Enc(v, k_{i,j}, m)$. $C_{i,j}$ 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, $C_{i,j}oC_{l,m} = C_{l,m}oC_{i,j}$.

B. The Basic Interleaved Encryption Protocol

This section describes the basic interleaved encryption protocol. It is assumed that node S wants to privately send a message m to node D. All the nodes along the path from S to 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.

It is also assumed that each node *only knows the next hop* on the path to the destination node D and that each node N_i on the route shares a pairwise key, $k_{i,i+1}$, with N_{i+1}. It is also assumed that N_i shares a key, $kn_{i,i+1}$ with N_{i+1}'s neighbors.

1) Protocol Description: The protocol executed by each node N_i, on the path from S to D, is described in Table I:

Interleaved Encryption Scheme (1)

- If (i==0) then N_S computes $c_0 = Enc(k_{s,1}, Enc(kn_{s,1}, m))$ and forwards c_0 to N_1.
- If $(i == D)$ then N_D computes $m' = Dec(kn_{D-2,D-1}, Dec(k_{D-1,D}, c_{D-1})$ and recovers the secret message that was sent by N_S.
- If (i==D-1) then N_i computes $t_i = Dec(kn_{i-2,i-1}, Dec(k_{i-1,i}, c_{i-1})$ and $c_i = Enc(k_{i,i+1}, t_i)$ and forwards c_i to N_{i+1}.
- Else N_i computes $t_i = Dec(kn_{i-2,i-1}, Dec(k_{i-1,i}, c_{i-1})$ and $c_i = Enc(k_{i,i+1}, Enc(kn_{i,i+1}, t_i))$ and forwards c_i to N_{i+1}

TABLE I

2) An example: This section gives an example of the previous algorithm using the network displayed on Fig. 2.

- The source node, S, selects two random Initialization Vectors, $iv_{S,1}$ and $iv_{S,2}$, computes $c_S = Enc(iv_{S,2}, kn_{S,1}, Enc(iv_{S,1}, k_{S,1}, m))$, and sends the message $\{D, S, iv_{S,1}, iv_{S,2}, c_S\}$ to node N_1 (i.e. the first node on the path toward D).
- Node N_1 computes $t_1 = Dec(iv_{S,1}, k_{S,1}, c_S)$ and retrieves $Enc(iv_{S,2}, kn_{S,1}, m)$ (note that since N_1 does not know $kn_{S,1}$, it cannot recover m). It then selects two random values, $iv_{1,2}$ and $iv_{1,3}$, computes $c_1 = Enc(iv_{1,3}, kn_{1,2}, Enc(iv_{1,2}, k_{1,2}, t_1))$ and sends the message : $\{D, S, iv_{S,2}, iv_{1,2}, iv_{1,3}, c_1\}$ to node N_2.
- N_2 computes $t_2 = Dec(iv_{1,2}, k_{1,2}, Dec(iv_{S,2}, kn_{S,1}, c_1))$ and retrieves $Enc(iv_{2,3}, kn_{1,2}, m)$ (since N_2 does not know $kn_{1,2}$, it cannot recover m). It then selects two random values, $iv_{1,2}$ and $iv_{1,3}$, computes $c_2 = Enc(iv_{2,4}, kn_{2,3}, Enc(iv_{2,3}, k_{2,3}, t_2))$ and sends the message: $\{D, S, iv_{1,3}, iv_{2,3}, iv_{2,4}, c_2\}$ to node N_3.
- ...
- Node N_i receives message $\{D, S, iv_{i-2,i}, iv_{i-1,i}, iv_{i-1,i+1}, c_{i-1}\}$ from N_{i-1}, computes $t_i = Dec(iv_{i-1,i}, k_{i-1,i}, Dec(iv_{i-2,i}, kn_{i-2,i-1}, c_{i-1}))$ and retrieves $Enc(iv_{i-1,i+1}, kn_{i-1,i}, m)$ (since N_i does not know $kn_{i-1,i}$, it cannot recover m). It then selects two random values, $iv_{i,i+1}$ and $iv_{i,i+2}$. If node N_{i+1} is node D, i.e. the destination node, node N_i computes $c_i = Enc(iv_{i,i+1}, k_{i,i+1}, t_i)$ and sends the message: $\{D, S, iv_{i-1,i+1}, iv_{i,i+1}, c_i\}$ to node N_{i+1}. If node N_{i+1} is not the destination, it computes $c_i = Enc(iv_{i,i+2}, kn_{i,i+1}, Enc(iv_{i,i+1}, k_{i,i+1}, t_i))$ and sends the message: $\{D, S, iv_{i-1,i+1}, iv_{i,i+1}, iv_{i,i+2}, c_i\}$ to node N_{i+1}.
- Node D (destination) computes $Dec(iv_{d-1,d}, k_{d-1,d}, Dec(iv_{d-2,d}, kn_{d-2,d-1}, c_d))$ and retrieves the original message m.

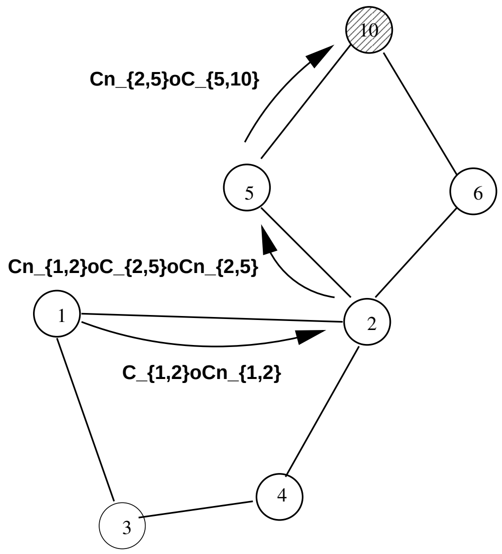

Fig. 2. **Interleaved Encryption**: Node 1 sends a message, m, securely to node 10. Node 1 encrypts m with $k_{1,2}$ and $kn_{1,2}$ and sends the results to node 2. Node 2 decrypts the message with $k_{1,2}$ and encrypts it with $k_{2,5}$ and $kn_{2,5}$. The results is sent to node 5. Node 5 decrypts the message with $k_{2,5}$ and $kn_{1,2}$, and encrypts it with $k_{5,10}$. The result is sent to node 10. Node 10 decrypts the message with $k_{5,10}$ and $kn_{2,5}$, and retrieves the initial message m.

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. The goal of an attacker is to decrypt and retrieve the message m.

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 [3], 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. However, as shown later in Section III-A, our scheme can in some situations be extended to become resistant to n-compromised nodes, where n is a system parameter.

Furthermore, 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, it received the packet from, 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) can not be used (even interleaved-authentication cannot be used [4], [5]). 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. If IV-based mode of operation is used, as in RC4, another solution is to derive the IV from a nonce and the destination address, for example $IV = hash(nonce, D)$. 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} from N_{i-1} equal to $Enc(iv^m_{i-1,i}, k_{i-1,i}, (Enc(iv_{i-2,i}, kn_{i-2,i-1}, Enc(iv^m_{i-1,i-2}, k_{i-1,i+1}, m))))$, where $iv^m_{j,l} = hash(nonce_{j,l}, D_m)$ and $iv_{j,l} = hash(nonce_{j,l}, D)$. Node N_i then computes $t^m_i = Dec(iv^m_{i-1,i}, k_{i-1,i}, Dec(iv^m_{i-2,i}, kn_{i-2,i-1}, c_i))$. Since node N_{i-2} used iv instead of iv^m, N_i will not retrieve $t_i = Enc(iv_{i-1,i+1}, kn_{i-1,i}, m)$ but $t^m_i = Dec(iv^m_{i-2,i}, kn_{i-2,i-1}, Enc(iv_{i-2,i}, k_{i-2,i}, Enc(iv_{i-1,i+1}, k_{i-1,i+1}, m))))$. Since $iv^m_{i-2,i}$ and $iv_{i-2,i}$ are different, t^m_i will differ from t_i. This error will then propagate until the destination node D_m and will scramble the ciphertext.. As a result, D_m won't be able to correctly decrypt the message and retrieve the correct plaintext m. The previous attack will fail.

As a result, this modified interleaved encryption scheme is secure against active attackers as long as no more than two adjacent nodes are not compromised.

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 [6].

It is believed that this combination of encryption and redundancy functions provide *plaintext integrity* (see remark 5.3 of [7]) 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 $\mathcal{F}$ [7]. Using the message length as an input of the redundancy function might also be a good design practice to avoid the attack described in [8].

D. Authenticated Interleaved Encryption

The section describes an extension of the basic protocol that provides integrity and privacy security against active attacks.

- The source node selects two nonces $nonce_{S,2}$ and $nonce_{S,1}$ and computes $iv_{S,2} = h(nonce_{S,2}||D||S)$ and $iv_{S,1} = h(nonce_{S,1}||D||S)$. It then generates $c_S = Enc(iv_{S,2}, kn_{S,1}, Enc(iv_{S,1}, k_{S,1}, m||R(m, len)))$, where $R(.)$ is a redundancy function and len is the length of message m. The message $\{D, S, nonce_{S,1}, nonce_{S,2}, c_S\}$ is sent to node N_1 (i.e. the first node on the path toward D).
- Node N_1 computes $iv_{S,1} = h(nonce_{S,1}||D||S)$, $t_1 = Dec(iv_{S,1}, k_{S,1}, c_S)$ and retrieves: $Enc(iv_{S,2}, kn_{S,1}, m)$. It then selects two nonces $nonce_{1,2}$ and $nonce_{1,3}$ and computes $iv_{1,2} = h(nonce_{1,2}||D||S)$ and $iv_{1,3} = h(nonce_{1,3}||D||S)$. It generates $c_1 = Enc(iv_{1,3}, kn_{1,2}, Enc(iv_{1,2}, k_{1,2}, t_1))$ and sends the message: $\{D, S, nonce_{0,2}, nonce_{1,2}, nonce_{1,3}, c_1\}$ to node N_2.
- N_2 computes $t_2 = Dec(iv_{1,2}, k_{1,2}, Dec(iv_{S,2}, kn_{S,1}, c_1))$ and retrieves $Enc(iv_{1,3}, kn_{1,2}, m)$. It then selects two nonces, $nonce_{2,3}$ and $nonce_{2,4}$, and computes $c_2 = Enc(iv_{2,4}, kn_{2,3}, Enc(iv_{2,3}, k_{2,3}, t_2))$. It sends the message $\{D, S, nonce_{1,3}, nonce_{2,3}, nonce_{2,4}, c_2\}$ to node N_3.
- ...
- Node N_i receives message $\{D, S, nonce_{i-2,i}, nonce_{i-1,i}, nonce_{i-1,i+1}, c_{i-1}\}$ from N_{i-1}, computes $iv_{i-1,i} = h(nonce_{i-1,i}||D)$, $iv_{i-2,i} = h(nonce_{i-2,i}||D)$ and $t_i = Dec(iv_{i-1,i}, k_{i-1,i}, Dec(iv_{i-2,i}, kn_{i-2,i-1}, c_{i-1}))$. It then retrieves $Enc(iv_{i-1,i+1}, kn_{i-1,i}, M)$.
  N_i selects two nonces, $nonce_{i,i+1}$ and $nonce_{i,i+2}$, computes $c_i = Enc(iv_{i,i+2}, kn_{i,i+1}, Enc(iv_{i,i+1}, k_{i,i+1}, t_i))$ and sends the message: $\{D, S, nonce_{i-1,i+1}, nonce_{i,i+1}, nonce_{i,i+2}, c_i\}$ to node N_{i+1}.

Note that if node N_{i+1} is the final destination D, node N_i computes $c_i = Enc(iv_{i,i+1}, k_{i,i+1}, t_i)$.

- Node D (destination) computes $Dec(iv_{d-1,d}, k_{d-1,d}, Dec(iv_{d-2,d}, kn_{d-2,d-1}, c_d))$ and retrieves the plaintext $x||m$. It accepts the message m if $x = R(m, len)$. Otherwise the message m is rejected.

III. Applications

The proposed AIE scheme is well adapted to devices, such as wireless sensors, with very limited computing and storage capabilities since it does not rely on public key cryptography and does not require a lot of storage. If d is the network's degree, i.e. the average number of neighbors each node has, $O(d^2)$ keys are required per node.

AIE can be used for many different applications. It can, for example, be used by a sensor to send secret data to nodes it does not share a security association with or that are defined by the network according to some criteria (location, functionality).

Another important application of AIE is short-lived broadcast encryption. With AIE, a node can send encrypted messages to very dynamic groups without having to establish and maintain group keys.

Finally, our scheme can be used together with the secure homomorphic stream cipher proposed in [9] to provide scalable secure aggregation of encrypted data in wireless sensor networks.

These applications are detailed in the rest of this section.

A. Securing Unicast Communication

AIE can be used by a sensor to send secret data to nodes it does not share a security association with. If the network uses a link state based routing scheme, a node only knows the newt hop to its final destination. In the case, the protocol described in Section II-B.1 can be used.

If the network uses a distance vector based routing scheme, the source actually knows all the nodes along the path to the destination. In this case, a more elaborate and more secure scheme can be used. This scheme is described in the rest of this section.

1) Key distribution model: In this scheme, we assume that each node do not only share a pairwise key with its direct neighbors but also with its $n - hop$ neighbors. Fig. 3 illustrates this key model. On this example, n is set to 2. Each node shares a pairwise key with each of its direct neighbors and each of its 2-hop neighbors. For example, node 1 shares a pairwise key with each of its direct neighbors, i.e. nodes 2, 3 and 4, and with its 2-hop neighbors, i.e. nodes 4, 5 and 6. These keys can be configured using a protocol such as the one described in [1].

2) Protocol description: The protocol, executed by each node N_i, is described by Table II. It is assumed that node N_0 wants to send a secret message to node N_D, and that N_1 is the first node on the path from N_0 to N_D, N_2

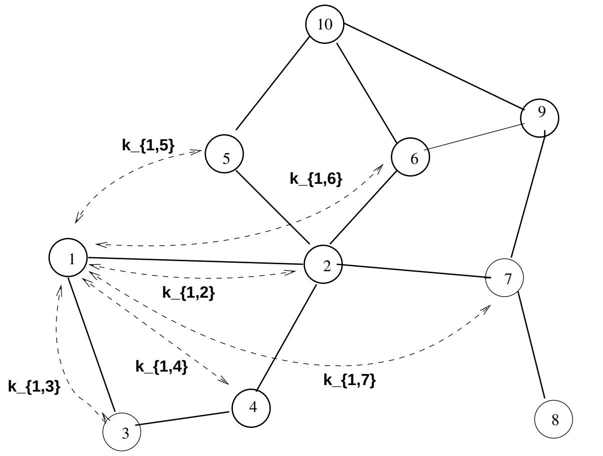

Fig. 3. **Key model**. Each node shares a pairwise key with each of its n-hop neighbors (n=2). This figures shows the keys that the node 1 shares with its direct neighbors and two-hop neighbors.

Interleaved Encryption Scheme (2)
- If $(i == 0)$ then N_0 computes $c_0 = Enc(k_{0,1}, Enc(k_{0,2}, ..., Enc(k_{1,n}, m)))$ and forwards c_0 to N_1. - If $(i == D)$ then N_D computes $m' = Dec(k_{D-n,D}, Dec(k_{D-n+1,D}, ...Dec(k_{D-1,D}, c_{D-1})))$ and recovers the secret message that was sent by N_0. - Else N_i computes $t_i = Dec(k_{i-n,i}, Dec(k_{i-n+1,i}, ...Dec(k_{i-1,i}, c_{i-1})))$ and $c_i = Enc(k_{i,i+1}, Enc(kn_{i,i+1}, ..., Enc(k_{i,i+n}, t_i)))$ and forwards c_i to N_{i+1}

TABLE II

the second node,...and N_D the destination node. It is also assumed that $k_{i,j} = 0$ if $i < 1$ or $j > p$.

This protocol is illustrated by the example of Fig. 4. In this example, node 1 sends a secret message m to node 10 using Interleaved Encryption (n=2). Node 1 encrypts m with $k_{1,2}$ and $k_{1,5}$ and forwards the result c_1 to node 2. Node 2 decrypts c_1 with $k_{2,1}$. It then encrypts the result with $k_{2,5}$ and $k_{2,10}$ and gets c_2. Upon reception of c_2, Node 5 decrypts it with $k_{2,5}$ and $k_{1,5}$. It then encrypts the result with $k_{5,10}$ and gets c_5. Upon reception c_5, Node 10 decrypts it with $k_{2,5}$ and $k_{5,10}$ and retrieve m.

3) Security Analysis: Since the message m is encrypted several times on each link, this protocol is secure against passive attacks.

It is also secure against active attacks as long as less than n consecutive nodes on the path do not get compromised. It is therefore more secure than the scheme described in Section II-B.1. However it requires each node on the path to know its n next nodes on the path to the destination. This information is not always available.

Note also that the integrity protection mechanism described in Section II-D can also be used to protect this protocol.

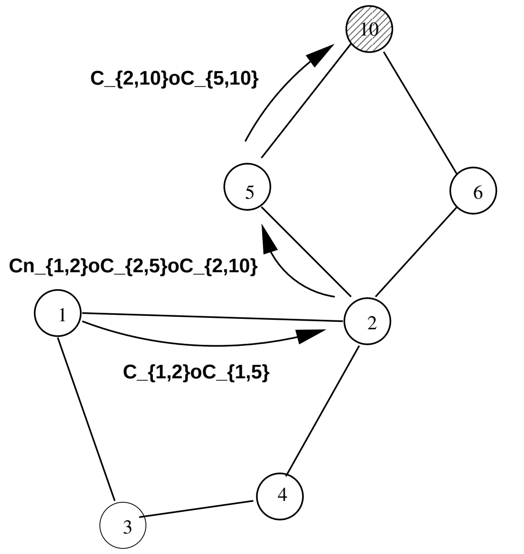

Fig. 4. **Secure Unicast using Interleaved Encryption**.

B. Dynamic Short-lived Multicast

One important service common to many sensor networks is the need to privatly 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 example2 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 definetly needed.

Protocol description: Our AIE 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-A.1 is used.

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 . 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).

  If node N_i has d neighbors, it has to perform $2 * (d-1)^2$ encryptions. We assume for simplicity that S sends the messages to all its neighbors, i.e. broadcast is used. However if a tree-based multicast scheme is used, the message is only sent to the neighbors that are on the delivery tree. Note that the messages sent to each neighbor are different since they are encrypted using different keys.
- The message then propagates along the delivery tree until it reaches all destination nodes.

Example1: Figure 5 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) towards the destination. In this case, the next node toward D is N_7. It then decrypts the message with $k_{1,2}$, and retrieves $Cn_{1,2}$ and encrypts 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 together with D. N_7 finds out that D_2 and $D3$ are its direct neighbors and that $D3$ is reachable via $D3$. 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. Identically 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 then retrieve m.

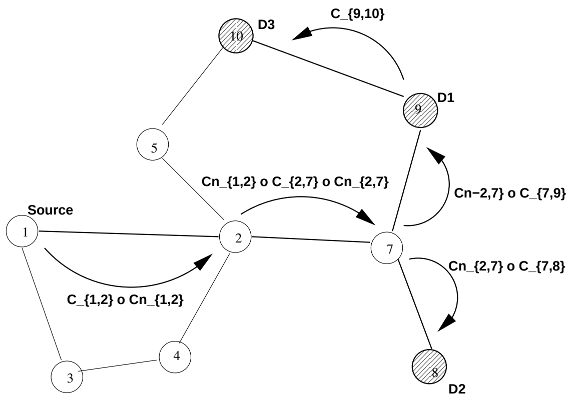

Fig. 5. **Short-lived Multicast**. Node N_1 multicasts a secret message to nodes N_8, N_9 and N_10.

Example2: Figure 6 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_7 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.

C. Aggregation of Interleaved Encrypted Data

1) background and problem statement: Wireless sensor networks (WSNs) are ad-hoc networks 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 life-

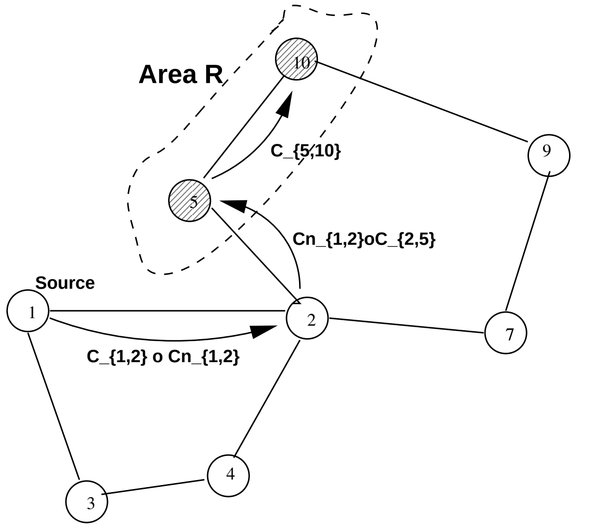

Fig. 6. **Location-Based Short-lived Multicast**. Node N_1 multicasts a secret message to all nodes with area R, i.e. nodes N_5 and N_{10}

time 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.

[9] proposes a simple and provably secure additively homomorphic stream cipher that allows efficient aggregation of encrypted data. This new cipher 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 [9] is to replace the xor (Exclusive-OR) operation typically found in stream ciphers with modular addition ($+$), as described by Table III.

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 $c = Enc(m, k, M) = m + k \pmod M$

Decryption:

1) $Dec(c, k, M) = c - k \pmod M$

Addition of Ciphertexts:

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

TABLE III

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 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 id. 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. In fact, a probabilistic encryption scheme, denoted $Enc_k()$, is additively-homomorphic if for any instance $Enc()$ of the encryption scheme, given $c_1 = Enc_{k1}(m_1)$ and $c_2 = Enc_{k2}(m_2)$, there exists a key k such that

$$c_1 \oplus c_2 = Enc_k(m_1 \oplus m_2)$$

In other words, the result of the application of function $\oplus$ on plaintext values may be obtained by decrypting the result of $\oplus$ applied to the corresponding encrypted values.

Additive aggregation can also be used to compute the variance, standard deviation and any other moments on the measured data. For example, in case of variance, each aggregator not only computes the sum, $S = \sum_{i=1}^{k} x_i$, of the individual values sent by its k children, but also the sum of their squares: $V = \sum_{i=1}^{k} x_i^2$. Eventually, the sink obtains two values: the sum of the actual samples which it can use to compute the mean and the sum of the squares which it can use to compute the variance.

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.

2) Our solution: Aggregation of Interleaved Encrypted Data : This section explains how the previous scheme can be used in an interleaved encryption mode to solve the identity transmission and scalability problems described in the previous section.

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. Fig. 7 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].

When a sensor, N_i, sends a message, its encrypts it n-time using the additively homomorphic scheme described in [9]. 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.

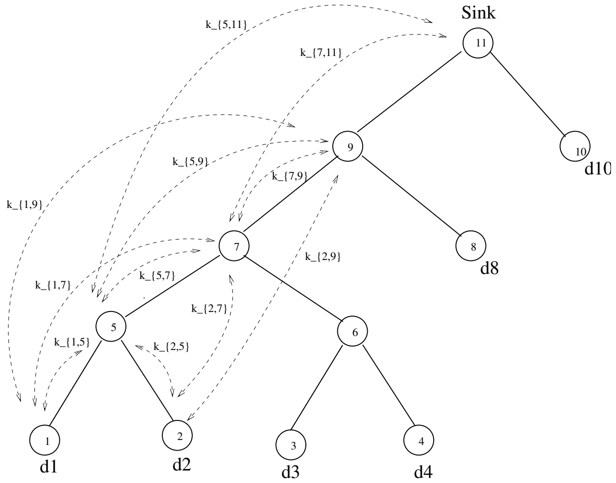

Fig. 7. **Aggregation of Interleaved Encrypted Data: Key model**.

Figure 8 illustrates secure aggregation with our interleaved encrypted data scheme. In this example, n is set to 3. 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

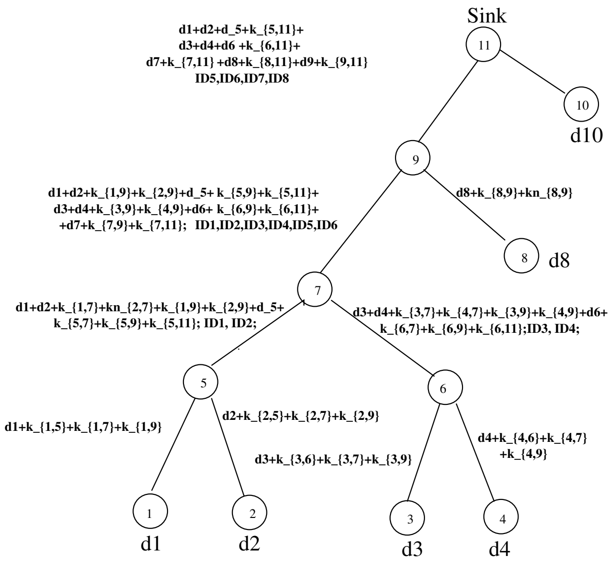

Fig. 8. **Aggregation of Interleaved Encrypted Data**. Data d_1, d_2, d_3, d_4 and d_8 are securely aggregated by the aggregators N_5, N_6 and N_7, on their way to the sink.

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 $d_1 + d_2 + d_3 + d_4 + d_5 + d_6 + d_7 + d_8 + d_9$.

3) Security vs Bandwith 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 contract, 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 Fig.8, 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 original scheme. An attacker that corrupts n consecutive nodes can actually retrieve the aggregated value at the lowest corrupted aggregator in the tree. For example, in Fig. 8 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 bandwith cost decreases but so does the security. 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 bandwith 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 [9]. Its bandwith cost is high but its security is maximum.

4) Data Integrity Protection: The authenticated scheme proposed in Section II-D 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 homomorphically additive. 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 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 [10], 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.

IV. 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. [11] improves and analyzes this scheme. Du et al. [12] extends the Eschenauer-Gligor scheme with a Blom pairwise key-generation scheme. Liu and Ning [13] 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 [14], [15], 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 [4] and Zha et al. [5] have proposed data integrity protection schemes for secure node-to-node communications based on interleaved authentication. Goodrich [3] 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.

V. Conclusion and Future Work

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.

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.

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, a 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, the AIE proposal is the first scheme to provide a solution to the challenging short-lived secure broadcast problem.

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

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

[5] 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.

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

[7] Hugo Krawczyk, "The order of encryption and authentication for protecting communications (or: how secure is SSL?)," in *CRYPTO*, 2001.

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

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

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

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

[12] 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.

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

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

[15] Yongdae Kim, Adrian Perring, and Gene Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *ACM Conference on Computer and Communications Security*, November 2000, pp. 235–244.