Secure and Practical Key Establishment for Distributed Sensor Networks

Panayiotis Kotzanikolaou[1], Emmanouil Magkos[2], Dimitrios Vergados[3] and Michalis Stefanidakis[2]

Department of Informatics[1]
University of Piraeus, 80, Karaoli & Dimitriou, 18534, Piraeus, Greece
pkotzani@unipi.gr

Department of Informatics[2]
Ionian University, Platia Tsirigoti, 49100, Corfu, Greece
{emagos,mistral}@ionio.gr

Department of Information and Communication Systems Engineering[3]
University of the Aegean, Karlovassi, Samos, GR-832 00, Greece
vergados@aegean.gr

February 5, 2008

Abstract

Key establishment in sensor networks is a challenging task, due to the physical constraints of sensor devices and their exposure to several threats. Existing protocols based on symmetric cryptography are very efficient but they are weak against several node impersonation and insider attacks. On the other hand, asymmetric protocols are resilient to such attacks but unfortunately, they are not feasible for sensor networks, even in their most efficient versions (*e.g.* the Elliptic Curve Diffie-Hellman family of key agreement protocols). In this paper we present two pairwise key establishment protocols for sensor nodes in unattended Distributed Sensor Networks (DSNs). The first protocol is hybrid and it combines asymmetric (Elliptic Curve) cryptography with symmetric key techniques. The second protocol is fully asymmetric. Furthermore, through simulations, we measure the efficiency of the proposed protocols in comparison with existing hybrid protocols. Our results show that under conditions, it is feasible for highly sensitive applications of static sensor networks to employ partial or fully asymmetric key establishment techniques and thus extend their security properties.

1 Introduction

Sensor nodes are small and inexpensive communication devices with limited storage, computation and energy capabilities. A typical sensor node is the

MICA2 mote, with 8-bit 8MHz processor, 128KB program memory, 4KB EEPROM and small size [8]. Sensor nodes are also able to communicate with each other with radio of limited bandwidth, thus forming a *Wireless Sensor Network* (WSN). In WSNs there is usually one or more base stations that connect the sensor nodes to the rest of the world. A base station is a computationally robust device that is usually assumed as being part of a trusted (secure) computing environment and that it cannot be easily compromised (*e.g.* [17, 28]). From a security perspective, of special interest are *self-organizing* WSNs where the BS does not take any part in the security mechanism of the network. Self-organizing WSNs may be subdivided into two distinct categories.

In *hierarchical sensor networks*, some nodes may be more powerful than other nodes, and/or play a different role than the rest of the nodes in the network. In a multi-level hierarchy for example, the network may be organized into clusters; in each cluster there is a clusterhead which may aggregate, process and forward information to the base station, or even take part in establishing link (path) keys within (or between) clusters (*e.g.*[27]). On the other hand, in *distributed sensor networks* (DSNs) there is not a fixed infrastructure and all nodes are equal in terms of sensing, routing and security capabilities. They also play their role in the network without external guidance or supervision. Schemes that follow this model assume that all nodes can be senders and receivers of a message.

1.1 Security in DSNs

The environment where the sensor nodes are deployed in the case of DSNs may be controlled (*e.g.* home, office) or uncontrolled (*e.g.* hostile territory). Of special interest is the case of unattended DSNs in hostile environments where all nodes are assumed physically accessible, and consequently more vulnerable. If a node gets compromised (for example by physical or logical attacks that extract all keys from its memory), the adversary can use these keys to launch several passive and active attacks against the network [6, 32]. For example, an adversary may try to eavesdrop or modify data and routing information, impersonate a node, or inject misleading information in the network. Moreover, detecting compromised nodes is very difficult. As a result, a challenge is to minimize the consequences of such compromise, especially in mission critical applications. As a minimal protection, compromising the cryptographic keys of a node at a given time, should not reveal past communications protected with keys used in past time, a property known as *forward secrecy* [15].

Key management in unattended DSNs is a difficult task when the network topology is also *dynamic*. If the environment is uncontrolled, deployment may have to be performed by randomly scattering the nodes into the target area. In DSNs with a dynamic topology, it is usually required that new nodes are able to join the network in future time, in order to replace the exhausted ones or simply to extend the network coverage. This can be achieved with *multiphase deployment* protocols (see for example [12, 35]). In such protocols, the incoming nodes are grouped in sets known as generations. Each generation of nodes joins the network in a specific time (phase) during which the newly arrived nodes are

engaged in a *bootstrap* protocol in order to exchange keys and to secure their communication with older nodes.

1.2 Our contribution

In this paper we present two pairwise key establishment protocols, suitable for DSNs with multiphase deployment. These protocols employ efficient asymmetric primitives, such as Elliptic Curve Cryptography and Implicit Certificates. Preliminary versions of these protocols have been presented in [18, 23]. Due to their public-key nature, the proposed protocols improve over the symmetric-key based schemes of [12, 35], as they do not allow a compromised node to impersonate other nodes belonging to the same or to a different generation. Furthermore, they provide forward secrecy both in respect to a particular node and a generation of nodes. Moreover, they do not require the assumption of a protected bootstrapping period, although if such a protection exists the security of the protocol is further increased. Finally, our protocols improve over the hybrid scheme of [16], since they support multiphase deployment, and do not require the existence of full-functional devices. Furthermore, we analyze the security properties and examine the feasibility of the proposed protocols in several environments. The efficiency analysis shows that the proposed protocols are scalable and efficient for low-capability static devices in terms of storage, communication and computational complexity: the cost per node for a key establishment is reduced to one scalar multiplication with a random point plus one with a fixed point.

2 Related work

Due to the resource constraints of sensor nodes and the lack of a fixed infrastructure, initial work for key management in DSNs focuses on symmetric-key solutions [1, 13, 32, 35, 12]. Such solutions are considerably more efficient for sensor nodes. However, they cannot provide adequate protection against impersonation attacks in highly sensitive applications (see Section 7).

2.1 Symmetric key-establishment protocols

In these schemes a number of symmetric keys are generated and pre-loaded into sensor nodes prior to their deployment. The degree of key sharing between the nodes of the system is varying.

In the case of *static shared key* protocols, a master global key is pre-shared throughout the network, or a unique link key is pre-shared between each pair of nodes in the network. A third approach is that the keys are pre-shared using deployment knowledge. In *global key pre-distribution* (*e.g.* [1, 32, 5]) all nodes of the network (or a group of nodes in the same vicinity cluster) are pre-deployed with a master network (or group) key. This solution is secure in controllable environments and implies low storage cost. In uncontrollable environments, if a

single node is compromised then the security for the entire network (or group) will be lost. In *pairwise key pre-distribution*, every node shares a secret key with every other node of the network, or with its immediate neighbors. This approach is perfectly resilient against node capture, but does not scale well and implies a prohibitive cost in storage for large networks. In key pre-distribution with *deployment knowledge* [11], keys are assigned given *a priori* knowledge of the position of a node, with a non-negligible probability. Similarly, such solutions are not suited to unattended DSNs with dynamic topology.

In the case of *random key pre-distribution* (*e.g.* [13, 7]), random ring of keys are pre-distributed to each sensor, and then a key discovery phase is run in order to establish pairwise keys and communication paths with each other. The existence of such paths is not certain but can be guaranteed with a non-negligible probability. Random key pre-distribution implies a trade-off between connectivity and resilience. Moreover, the size of the key ring is increased with the network size. Another case is *polynomial-based key pre-distribution* (*e.g.* [2, 22, 29]). A polynomial share is distributed to each node, and using this share, every pair of nodes is able to establish a link key. Solutions of this category have a security parameter t, where a coalition of less than $t + 1$ nodes knows nothing about the pairwise keys of other nodes.

Finally, with *pairwise key establishment* the sensor nodes can use pre-deployed keying material in order to establish fresh link keys with their neighbors [4]. After having these link keys in place, it is possible to find a trusted *path* between any two nodes of the network. In the symmetric setting, the pre-deployed material can be a group key or a network-wide master key [12, 35].

2.2 Key establishment protocols based on asymmetric cryptography

Recent findings have demonstrated that, under conditions, public key cryptography may be used for pairwise key establishment in WSNs [14, 16, 24]. While it is impractical to use traditional public key cryptography or Key Distribution Centers for establishing link keys between nodes, *Elliptic Curve Cryptography* (ECC) can offer equivalent security with substantially smaller keys *e.g.*, a 160-bit key is expected to offer comparable security with an RSA 1024-bit key. Well known implementations of ECC for key agreement, such as the MQV family of protocols ([25, 21, 20, 19]) among others, are not optimally efficient, as they involve expensive public key operations for explicitly verifying public key certificates and computing mutually agreed keys.

Huang *et al* [16] and Kotzanikolaou et al [18] proposed *hybrid* protocols by combining ECC and symmetric techniques to perform pairwise key establishment between two neighboring nodes. The model of [16] suits hierarchical WSNs well, since it assumes some fully functional devices that take most of the cryptographic burden. The [18] model assumes that all nodes are equal, thus it is conceivably more suitable to unattended DSNs with dynamic topology.

Both [16] and [18] are hybrid in that they replace some expensive public-key operations with efficient symmetric ones. More specifically in [18], all nodes

belonging to the same group are pre-deployed with a (long-term) public key pair and a symmetric group key. This group key is later used during an Elliptic Curve Diffie Hellman (ECDH) [31] key establishment, to set a secure channel for contributing some freshness to the ECDH key. The fully asymmetric scheme of [23], addresses a weakness of the [18] scheme, which allows an adversary to learn past communications of a compromised node. The new scheme has similar communication and computation costs. Its improved security comes at an increased storage cost for each node. A full performance analysis for the asymmetric key establishment schemes will be given in Section 7.

2.3 Multi-phase DSN (MDSN)

In [17], newly arrived sensors are introduced to the network with the Kerberos-like approach of using a Base Station as an intermediate online server. Other schemes [35, 12], employ symmetric techniques to establish secure links between sensors that are deployed in different phases (or, generations), but are not adequately resilient against node capture [18]: if a node is compromised during the bootstrapping period, then the corrupted node can present multiple identities to other nodes of (the same) generation, an attack also known as a *sybil attack* [10]. Furthermore, any compromised node belonging to a generation i is able to pretend to newly deployed nodes that it belongs to a generation $j > i$ and establish a secure channel with them. This attack has been referred to as a *fake generation attack* [18]. The fully asymmetric scheme [23] addresses most security issues in unattended DSNs with multi-phase deployment. In Section 6, a security analysis of the schemes in [18] and [23] will be presented.

3 Design goals

We will first describe the design goals for secure and efficient key establishment in DSNs. Then we will describe a generic solution for asymmetric-key based solutions as well as efficient implementations of the generic solution, suitable for DSNs.

We assume that the network topology is dynamic and cannot be known in advance. Sensor nodes are pre-deployed with some initial keying information and credentials by a Trusted Authority (TA). The authority is trusted on issuing correct user public data for the reconstruction of the implicitly certified public keys. The authority is also trusted on not disclosing the long-term private keys of the sensor nodes. We further assume that the underlying cryptographic algorithms will not be broken.

3.0.1 Security goals and assumptions

We assume a network of sensor nodes that is deployed in a hostile territory. We describe several cryptographic assurances and traditional security goals, common to any system for key agreement in open networks, and elaborate in the context of a MDSN.

1. Cryptographic assurances. The protocol has to offer authenticated key agreement. Authentication should be mutual, while messages and keys exchanged should be fresh. Any established keys should not be totally controlled by any party. Both nodes should also obtain key confirmation, *i.e.*, assurance that the other node knows the agreed session key. The TA is trusted on certifying public keys and on not disclosing long term secrets that nodes are pre-deployed with.

2. Security against attacks. We assume both passive and active attackers against the system. We follow the Dolev-Yao threat model [9]. The attacker will eavesdrop, modify and replay messages in the protocol, steal session keys and/or long-term keys, generation-wide keys and cryptographic parameters, attempt to impersonate a node in a network or falsely claim to belong to a given generation of nodes. We do not deal with denial of service (DOS) attacks at the lower layers, which, can be orthogonally addressed [34].

3. MDSN-specific security. We further elaborate on a set of requirements that are specific to multiphase deployment sensor nodes:

   - Known-key security per node. Compromise of past session keys should not allow compromise of future session keys or impersonation of a sensor node.
   - Known-key security per generation. Compromise of generation-wide keys should not allow compromise of future session keys or allowing fake generation attacks.
   - Perfect forward secrecy [15] per node and per generation. Compromise of long term secrets (symmetric or asymmetric) should not compromise past session or generation-wide keys.

3.1 Efficiency goals

- Communication. The protocol should have minimal communication rounds and the total transmitted bits should be kept low.
- Computation. The complexity of computation should be minimal. If possible, pre-computation of several values can be used to reduce the total computation time.
- Storage. The storage requirements of the key establishment protocol should be minimal, regardless of the number of node generations. Since the storage cost depends on the number of node generations, optimally, the cost should increase logarithmically to the number of node generations.

4 A generic solution for asymmetric key establishment

We will first present a generic protocol for multiphase key establishment in DSNs, which combines public key and symmetric cryptography. Then we will discuss specific implementations by using well known practices.

We assume that during a setup stage, each node A is pre-deployed with the following keying material:

- The public key reconstruction data PK_A^{inf}, which will be used by other nodes to implicitly confirm a certified public key of node A. Each node is also pre-deployed with an authentic copy of the public certification key of the TA.
- A private/public key pair PK_A, SK_A, to be used for entity authentication as well as for key agreement. The mutually agreed key with any other node B will be a secure function of their static key pairs (*e.g.* as in the Diffie-Hellman protocol). As will be shown in the protocol, a set of random nonces R_A, R_B will also be exchanged in order to be used as a "salt" for computing the final session key.
- Symmetric cryptographic material $Crypt_N$ for network-wide authentication.
- Symmetric cryptographic material $Crypt_j$ to be used for intra-generation authentication (*i.e.* within generation j). $Crypt_j$ can also be used as a key-encrypting key for computing session keys with other nodes of the same generation j, as well as with nodes belonging to an older generation. For secure inter-generation communication with nodes belonging to a future generation i, where $i > j$, the following mechanism will be used [35]; during setup, the TA will bind, using a secure cryptographic transformation, the key $Crypt_i$ with the identity of the node A, thus producing credential $Crypt_i^A$. The transformation should not make it easy to compute $Crypt_i$ from $Crypt_i^A$, or to replace A with another identity. Of course, a newly arrived node $B^{(i)}$ (*i.e.*, belonging to generation i) will be able to compute $Crypt_i^A$ for an older node A.

Figure 1 presents a generic key agreement protocol for multiphase deployment DSNs. The schematic representation covers both cases of *intra-generation* communication (*i.e.* the nodes $A^{(j)}$ and $B^{(i)}$ belong to the same generation and $i = j$), as well as of *inter-generation* communication (*i.e.* the nodes $A^{(j)}$ and $B^{(i)}$ belong to different generations and $i > j$).

At a high level, each pair of nodes exchange some public key reconstruction data that will be used to compute the public keys for the key agreement protocol. Furthermore, each node encrypts and sends some randomness in order to "salt" the mutually agreed key. Throughout the protocol, both nodes authenticate their messages with the pre-deployed cryptographic credentials and

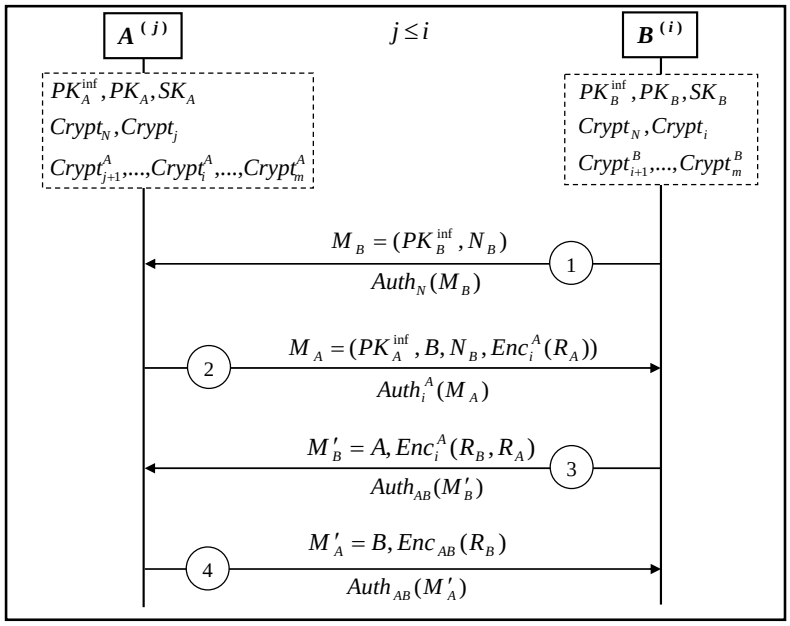

Figure 1: A generic key agreement protocol for multiphase DSNs

prove possession of the established key by encrypting and/or authenticating the exchanged messages. Observe that random values R_A, R_B, serve two purposes in the protocol. First, they serve as challenges for mutual entity authentication. Second, they serve as a salt to mutually agreed key. By using a suitably chosen key derivation function, both nodes will be able to compute the same session key.

At the end of the bootstrapping phase for the reference period i, and when most nodes have performed intra-generation and/or inter-generation handshakes with their neighbors, network nodes are programmed to delete all generation-wide credentials $Crypt_i$, as well as credentials of the type $Crypt_i^X$ that were pre-stored in an older node X or computed by a newer node that possesses $Crypt_i$.

Note that the generic protocol can be suitable for key establishment in multi-phase deployment DSNs, if the number and cost of public key operations is kept low. In the next sections, we present efficient implementations of the generic scheme.

5 Efficient implementations of key establishment protocols for multiphase deployment DSNs

We describe two implementations of the generic scheme for MDSNs, namely a *hybrid* and a *fully asymmetric* protocol. Both protocols are based on Elliptic Curve Diffie-Hellmann (ECDH) key establishment [31]. Furthermore, *Implicit Certificates* [33] are used to establish the authenticity of public encryption keys.

Both protocols involve a *key pre-deployment phase* and several *key bootstrapping phases*. The key pre-deployment phase is executed before the initiation of the network. Each sensor node is pre-deployed off-line by a trusted authority CA with the appropriate keying material, which will be later used for pairwise key establishment. The CA has no further involvement in the key establishment protocol. Then, the nodes of the first generation are randomly deployed in the area and the first bootstrapping phase is executed. This enables the initial nodes to establish unique pairwise keys. A new bootstrapping phase is executed every time a new generation of nodes arrives in the network, in order to enable the incoming nodes to establish pairwise (link) keys with each other, as well as with the existing nodes. After two neighbouring nodes establish a link key, no public key operations are needed: a link key may be updated using purely symmetric techniques *i.e.* the link key could be hashed to derive the next link key.

The hybrid protocol combines ECDH with Implicit Certificates and generation-wide symmetric encryption keys. The generation key is used at the beginning of the key boostrapping phase: every two nodes that belong to the same generation, create a temporary (weakly) secure channel in order to exchange some randomness for a "fresh" ECDH key. At the end of each key bootstrapping phase the respective generation-wide symmetric key is deleted.

The fully asymmetric key establishment protocol is basically a modified version of the previous protocol. It also combines ECDH with Implicit Certificates. However, this protocol is fully asymmetric and does not use any generation-wide symmetric keys in order to exchange randomization. This leads to the generation of static ECDH pairwise keys between each pair of nodes. Each node is pre-deployed with multiple public/secret Elliptic Curve (EC) key pairs, one for each bootstrapping phase. During a specific phase, each node uses its corresponding EC key pair in order to establish a key with every other node. At the end of the bootstrapping phase, the corresponding EC key pair is deleted.

For the rest of this section, we first describe in detail the proposed hybrid key establishment protocol. Then, we describe the fully asymmetric protocol, based on its differences with respect to the hybrid version.

5.1 The hybrid protocol

Let q denote the order of the underlying finite field F_q and let E be a suitably chosen elliptic curve defined over F_q. Let P denote a base point in E, the generator point, and n be the order of P, where n is prime. Thus $nP = O$ and $P \neq O$ where O is the point at infinity. We assume that the discrete

logarithm problem in the group $< P >$ of points generated by P is intractable. Let $q_{CA} \in [2, n-2]$ be a random integer selected by the Certification Authority CA and $Q_{CA} = q_{CA} \times P$. The pair of the static secret/public key pair of the CA is q_{CA}, Q_{CA}.

5.1.1 The key pre-deployment phase of the hybrid protocol

The CA generates a network-wide symmetric key K, which will be used by all nodes as an initial authenticator in order to avoid processing of fake "hello" messages and prevent trivial DoS attacks. Furthermore, the CA also generates a set of independent symmetric encryption keys, $K_1, K_2, ..., K_m$, one key for each of the m node generations. These keys are similar to the generation keys of the LEAP protocol [35]; however, in the hybrid protocol these keys will only be used to create a temporary channel in order to exchange randomness for the key establishment.

The CA generates and pre-deploys each node with the appropriate keying information (see Figure 2). We describe the key generation and key predeployment phase for a node X of generation i, denoted as $X^{(i)}$. When no further clarification is required, we will denote the node $X^{(i)}$ as X. The CA selects a random number $g_X \in [2, n-2]$ and computes $G_X = g_X \times P$. Then, the CA computes the Implicit Certificate for the node X as $IC_X = (G_X, M)$, with $M = \{i, ID_X, t_X\}$, where i is the generation of the node, ID_X is a unique identifier for the node X and t_X is the expiration time of the certificate. The CA applies a cryptographic hash function h over IC_X and from the octet $h(IC_X)$ it obtains an integer e_X, by using the conversion routine[1] described in [31]. Then, the CA computes the secret EC key of node X as $q_X = g_X + e_X \cdot q_{CA}$. The value g_X is not given to the node X and is deleted after the key generation process. Otherwise, a compromised node would be able to extract the secret key of the CA from values q_X and g_X. Observe that the pair (e_X, q_X) is an EC-Schnorr signature [30], created by the CA, over the message M of the Implicit Certificate IC_X of the node X. The corresponding public EC key Q_X is not stored at node X's memory. Any other node, will be able to recover Q_X from the implicit certificate IC_X and the public key Q_{CA} of the CA.

After the computation of the public/secret key pair of the node X, the CA computes the secret symmetric key values of the node. Since the node X belongs to the i_{th} generation ($1 \leq i \leq m$), the node will be given the corresponding generation-wide key K_i. Furthermore, for all future generation keys, the CA will compute for each node X the instance keys $K_{i+1}(ID_X) = f_{K_{i+1}}[ID_X]$, ..., $K_m(ID_X) = f_{K_m}[ID_X]$, where f is a one-way keyed hash function. Finally, the CA pre-deploys the node X with its secret EC key q_X, the Implicit Certificate IC_X, the public key of the Certificate Authority Q_{CA}, the point P, the initial authentication key K, the generation-wide key of the i_{th} generation K_i and the instance keys $K_{i+1}(ID_X), K_{i+2}(ID_X), ..., K_m(ID_X)$.

[1] Informally, the idea is simply to view the octet string as the base 256 representation of the integer (Section 2.3.8 of [31])

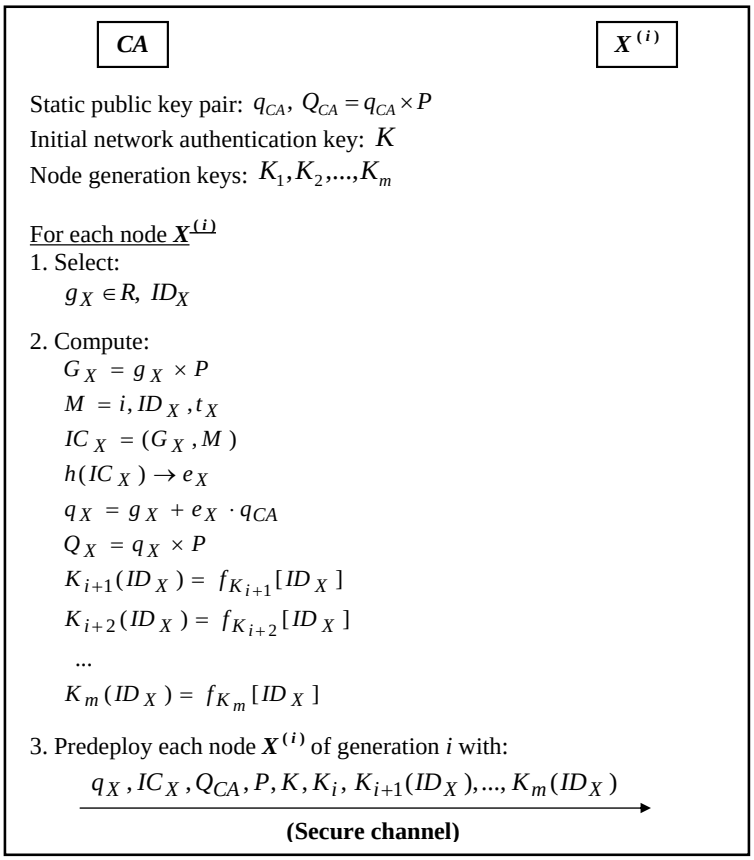

Figure 2: The key pre-deployment phase of the hybrid protocol

5.1.2 The key establishment phase of the hybrid protocol

In this phase, two nodes will use their pre-deployed keys to perform an authenticated pairwise key establishment. There are two cases to be considered: key establishment between nodes of the same generation and key establishment between nodes of different generations.

Let $A^{(j)}$, $B^{(i)}$ be two nodes belonging to the generations j, i respectively, such that $1 \leq j \leq i \leq m$. Thus, the nodes may belong to the same $(j = i)$ or different $(j < i)$ generation. We describe the key establishment phase of the i_{th} period (see Figure 3).

Both nodes will possess, among others, the i_{th} generation key or an instance of that key: If $j = i$, then $K_j = K_i$ and both nodes will possess the key K_i. Otherwise, if $j < i$, the node of the preceding generation $A^{(j)}$ will not possess the key K_i of the i_{th} generation. Instead, it will have already been pre-deployed with the instance $K_i(ID_A)$ of the i_{th} generation key K_i.

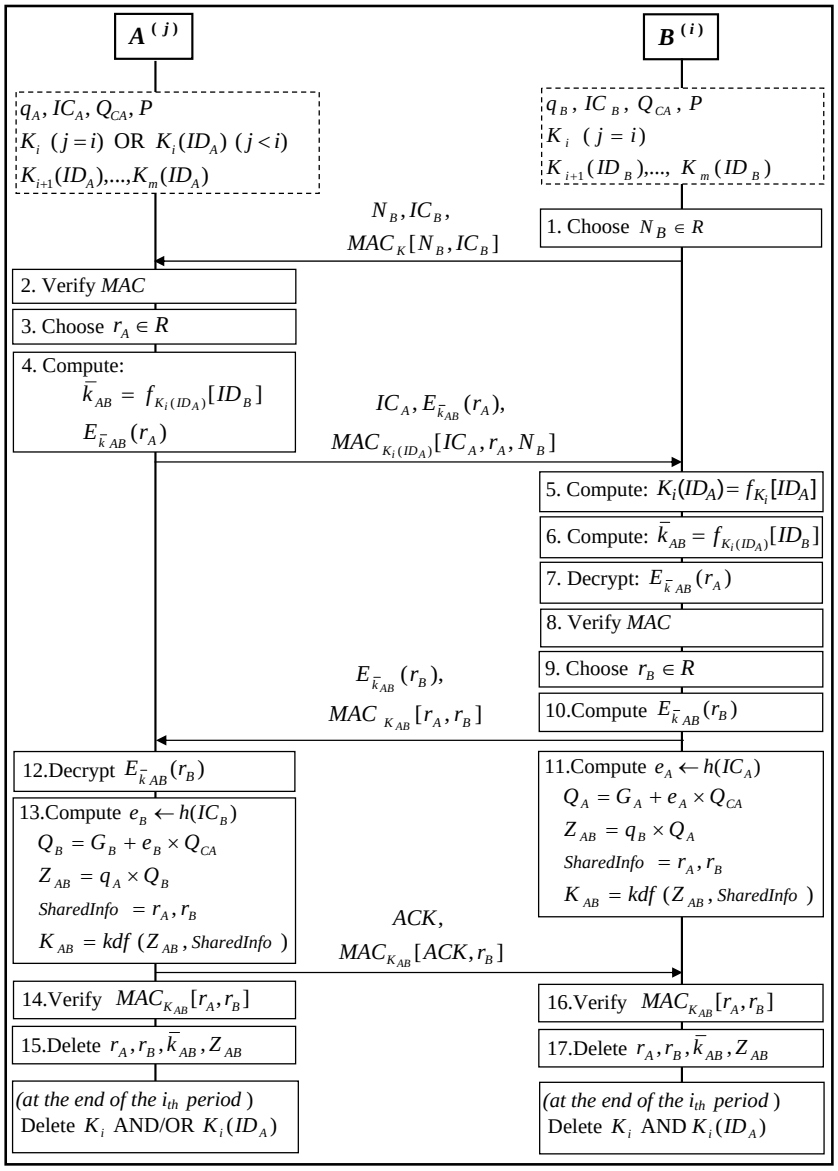

Figure 3: Key establishment phase of the hybrid protocol

Step 1. The node $B^{(i)}$ initiates key establishment, by choosing a random nonce N_B and broadcasting this along with its Implicit Certificate IC_B. For the initial authentication of the key establishment, the node B also broadcasts a Message Authentication Code (MAC) of the above values, generated with the initial au-

thentication key K.

Steps 2-3. The neighboring node A receives the MAC and verifies it. If the verification succeeds, it chooses a random number r_A, which will be used in the randomization of the ECDH key exchange.

Step 4. In order to protect the random value from eavesdroppers, the node A will generate a temporary key $\bar{k}_{AB}$ and encrypt r_A with that key. The temporary key $\bar{k}_{AB}$ is generated as follows. If both nodes belong to the same generation ($j = i$) then both nodes possess the generation key K_i. In that case, both nodes can generate the key $K_i(ID_A)$. If A is a node of a previous generation ($j < i$), then the node $A^{(j)}$ will have been pre-deployed with the key $K_i(ID_A)$. In both cases, from the key $K_i(ID_A)$, the node A can compute the temporary key as $\bar{k}_{AB} = f_{K_i(ID_A)}[ID_B]$. Then, the node A sends the encryption $E_{\bar{k}_{AB}}(r_A)$ to B, along with its Implicit Certificate IC_A and a MAC on IC_A, r_A, N_B generated with the key $K_i(ID_A)$.

Steps 5-8. On receiving this message, the node B computes the key $K_i(ID_A)$, by using its generation key K_i. Then, B uses $K_i(ID_A)$ to compute the temporary key $\bar{k}_{AB}$, and decrypts $E_{\bar{k}_{AB}}(r_A)$ to obtain r_A. Finally, B verifies the received MAC before proceeding to the next step.

Steps 9-10. The node B also chooses a random value r_B that will be used in the pairwise key establishment and encrypts it with the temporary key $\bar{k}_{AB}$.

Step 11. At this time, the node B will use the received Implicit Certificate IC_A and the public key Q_{CA} of the CA, in order to compute the public key of node A as $Q_A = G_A + e_A \times Q_{CA}$. Observe that at this point B cannot yet establish that Q_A is authentic: as soon as A proves knowledge of q_A, the node B will have *implicit* [33] assurance that it is talking to A and that all information included in the certificate is genuine (*i.e.* signed by the CA). The node B computes the static pair key $Z_{AB} = q_B \times Q_A$. The final pairwise key K_{AB} is computed by applying a key derivation function kdf over Z_{AB} and $SharedInfo$, where[2] $SharedInfo = r_A, r_B$. Thus, $K_{AB} = kdf(Z_{AB}, SharedInfo)$. The function kdf is implemented through an one-way cryptographic hash function, such as SHA-1. Then, the node B computes a MAC on r_A, r_B with the pairwise key K_{AB} and sends $E_{\bar{k}_{AB}}(r_B)$, $MAC_{K_{AB}}[r_A, r_B]$ to the node A. The MAC will provide key confirmation to node A, since it will prove that the corresponding secret key q_B was used.

Steps 12-13. The node A decrypts $E_{\bar{k}_{AB}}(r_B)$ and obtains r_B. Then the node A will use the Implicit Certificate IC_B and the public key Q_{CA}, in order to compute the public key of node B as $Q_B = G_B + e_B \times Q_{CA}$. Finally, the node

[2] In standard $ECDH$ [31], $SharedInfo$ is an optional string including some mutually known private information (specified as $suppPrivInfo$).

A computes the static pair key $Z_{AB} = q_A \times Q_B$. The pairwise key is again computed as $K_{AB} = kdf(Z_{AB}, SharedInfo)$, where $SharedInfo = r_A, r_B$.

Steps 14-16. Now the node A will verify the received MAC in order to confirm that the appropriate secret key of node B was used in the computation of K_{AB}. At this point the node A is assured about the authenticity of Q_B. In order to provide key confirmation regarding its own secret key q_A, the node A will also compute a MAC with the key K_{AB} and send it to node B.

Steps 15-17. After the MAC verification, both nodes will delete the random values r_A, r_B, the temporary key $\bar{k}_{AB}$ and the static key Z_{AB}. The nodes will then use the pairwise key K_{AB} for the actual communication. Note that from the key K_{AB} the two nodes can derive two different keys, one for encryption and one for authentication [31].

At the end of the i_{th} bootstrapping phase and after the nodes have performed a key establishment with each of their neighbors, they will delete the generation key K_i and/or the keys $K_i(ID_A)$, $K_i(ID_B)$ they possess. In the next bootstrapping phase the node A (respectively B) will use its secret static key q_A (resp. q_B) as well as its instance of the next generation's key $K_{i+1}(ID_A)$ (resp. $K_{i+1}(ID_B)$) in order to participate in the bootstrapping phase with the nodes of the generation $i + 1$.

5.2 The fully asymmetric protocol

As described earlier, the fully asymmetric protocol is a modified version of the hybrid protocol. Thus, we describe the differences between the hybrid and the fully asymmetric. The notation used is essentially the same as in the hybrid protocol.

5.2.1 The key pre-deployment phase of the fully asymmetric protocol

In the fully asymmetric protocol, there are no generation-wide symmetric keys $K_1, K_2, ..., K_m$. Only the initial network-wide symmetric key K is used. Instead of the generation-wide symmetric keys, in this protocol each node $X^{(i)}$ uses multiple independent EC key pairs, one EC key pair for each key bootstrapping period. The CA pre-deploys each node $X^{(i)}$ belonging to the i_{th} generation with $(m - i + 1)$ asymmetric EC key pairs (q_{x_k}, Q_{x_k}), $k = i, i + 1, ...m$ and their corresponding Implicit Certificates. In each forthcoming key bootstrapping phase $k = i, i+1, ...m$ the node $X^{(i)}$ will use the corresponding k_{th} key pair for key establishment with any node $Y^{(k)}$, belonging to the k_{th} node generation. The rest of the key pre-deployment phase is the same as in the hybrid protocol (see Figure 4).

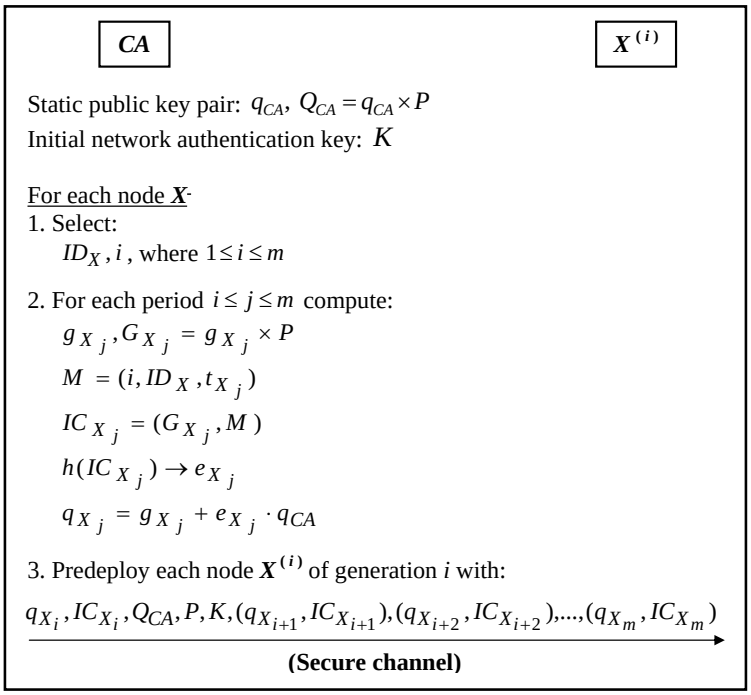

Figure 4: The key pre-deployment phase of the fully asymmetric protocol

5.2.2 The key bootstrapping phase of the fully asymmetric protocol

In each bootstrapping phase, any pair of sensor nodes lying in each other's range, will use their pre-deployed EC keys which correspond to the current bootstrapping period, in order to perform an authenticated pairwise key establishment, as shown in Figure 5. As in the previous protocol, we describe a key establishment between two nodes $A^{(j)}$, $B^{(i)}$, where $1 \le j \le i \le m$ (*i.e.* either both nodes are of the same generation or the node A belongs to a previous node generation).

Steps 1-3. These steps involve the initiation of the key establishment and are essentially the same as in the hybrid protocol.

Steps 4-6. The nodes B, A use the received Implicit Certificate IC_{A_i}, IC_{B_i} respectively, the public key Q_{CA} of the CA, and their current secret EC keys q_{B_i}, q_{A_i} respectively, in order to compute the static pair key $Q_{AB_i} = q_{B_i} \times Q_{A_i} = q_{A_i} \times Q_{B_i}$. The final pairwise key K_{AB} is computed by applying a key derivation function *kdf* over Q_{AB_i} and N_A, N_B.

Steps 7-10. Both nodes use the derived key and exchange a MAC generated with the derived key for key confirmation. If the confirmation succeeds, then the nodes accept the derived key as their pairwise key and delete the nonces

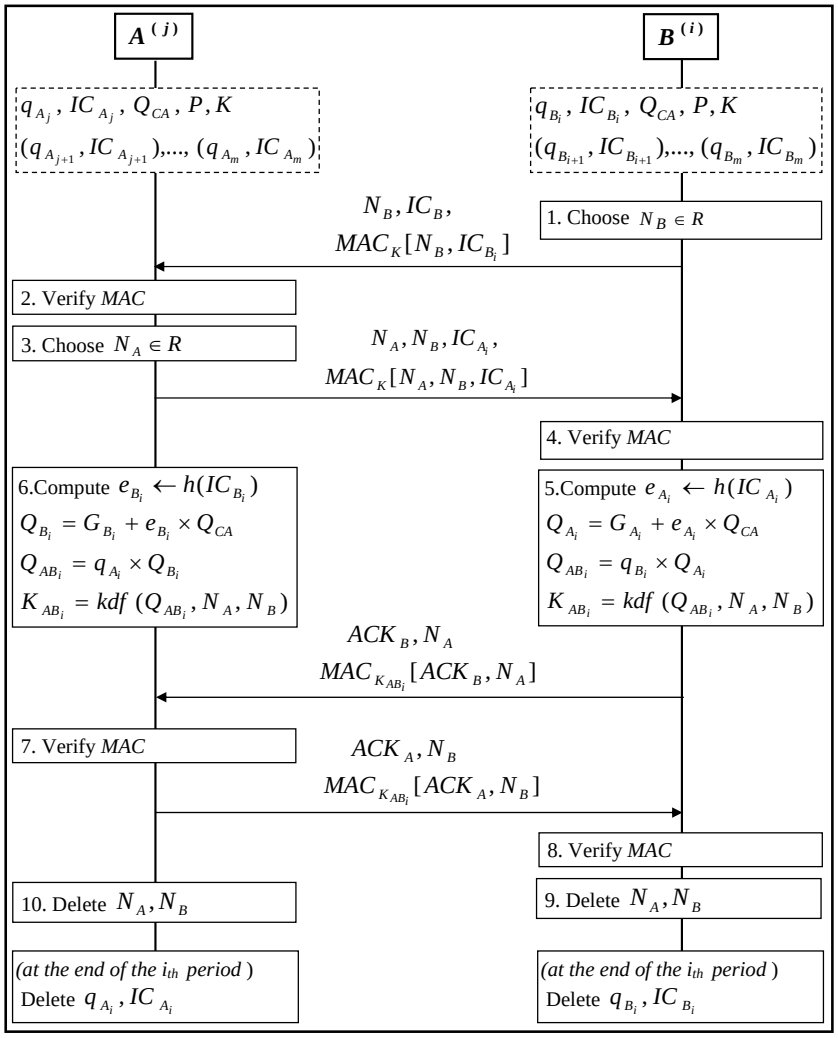

Figure 5: The key bootstrapping phase of the fully asymmetric protocol

N_A, N_B.

At the end of the i_{th} bootstrapping phase and after the nodes have performed the required key establishments, the nodes A and B will delete their i_{th} EC keys and certificates (q_{A_i}, IC_{A_i}), and (q_{B_i}, IC_{B_i}) respectively. For key freshness in subsequent periods, the nodes may periodically update the pairwise key K_{AB_i} using an one-way hash function.

Remark. Consider two nodes which are not able to communicate during the i'th bootstrapping phase (*e.g.* due to a weak signal, or a node being busy).

In the proposed protocol they will be able to establish session keys in a future bootstrapping phase by using the corresponding public key pair. This is not possible in the hybrid protocol where all nodes deleted their generation-wide keys at the end of each bootstrapping phase.

6 Security of the proposed protocols

6.1 General

The aforementioned protocols contain a four-pass challenge-response mechanism for authenticated key agreement with explicit key confirmation. They combine standard ECDH key agreement [31] and implicit certificates [33] for mutual authentication. In the hybrid protocol the nodes also share some a priori secret information. The authenticity of the implicit certificates is based on EC-Schnorr [30] signatures, which are provably secure under the *random oracle* model given that the discrete logarithm problem over a subgroup $< G >$ is untractable [3]. Both protocols make use of random nonces for message freshness, symmetric encryption for data confidentiality and MACs for data integrity. We assume that the underlying primitives are secure.

6.2 Secure key generation

By using a private offline interface between each sensor node and the CA, during pre-deployment phase, both active and passive attacks against the key generation process (such as unknown key share attacks and small subgroup attacks [20]) are thwarted, provided that the CA is honest and takes all reasonable measures in the key generation process.

6.3 Known key security

Clearly, if a node's keying material is revealed at any time, all its present and future communication is revealed, given that the attacker is also an eavesdropper. In the following we will examine whether the forward secrecy property is maintained throughout the execution of the protocols.

Forward secrecy per node (The hybrid protocol). The computation of a session key K_{AB} is based on both the static ECDH key Z_{AB} and the random values r_A, r_B. At the end of a key bootstrapping phase, both nodes delete the random values r_A, r_B, as well as the temporary encrypting keys $K_i(ID_A)$, $\bar{k}_{AB}$ which were used to exchange r_A and r_B. In addition, at the end of each bootstrapping phase, newly arrived nodes belonging to any generation i, delete their generation-wide key K_i. Consequently, the hybrid protocol protects past communication of a compromised node $A^{(i)}$, as long as an attacker does not have access to *both* the static EC private key q_{A_i}, and the generation-wide key K_i.

An attack on the hybrid protocol. Consider an attacker, who also eavesdrops on communication lines. The attacker succeeds in learning all generation-wide keys, *e.g.* by compromising at least one newly arriving node in each bootstrapping phase. Then, for any node which is compromised, during the life of the network, the attacker will be able to learn all node's past and future communications with any other node of the network [23].

Forward secrecy per node (The fully asymmetric protocol). The derivation of a session key K_{AB_i} is based on the static EC keys q_{A_i} and q_{B_i}, and the random values N_A, N_B. Since N_A, N_B are transported in the clear, they do not add security to the protocol. However, the nodes A, B delete their EC keys q_{A_i}, q_{B_i} respectively, at the end of the i'th bootstrapping phase and after they have performed a key establishment with each of their neighbors. Recall that in subsequent periods, the nodes will update the pairwise key K_{AB_i} using an one-way hash function. Thus, compromising a node at a given time does not reveal past communications of the attacked node. This would require knowledge of the particular EC keys used for any past session key.

Forward and backward secrecy per generation. Stealing all the keys of a node does not compromise past or future communications of any other node of the same or of a different generation. In both protocols, a session key for each pair of nodes is a function of the private EC keys of the nodes. Thus, the compromisation of a given node will not reveal past of future communication keys of any other node of the network.

6.4 Security against impersonation attacks

Since compromise of a node cannot be prevented, an attacker who compromises a node is able to impersonate this node and join in a communication with any other node of the network. In the proposed protocols, the attacker can do so only for the compromised node. However the communication between any other non-compromised nodes of the network remains secure (as opposed to symmetric key establishment protocols).

To prevent impersonation attacks both protocols make use of implicit certificates [33]. During a bootstrapping phase, the node A uses the implicit certificate of B and the public key of the CA to reconstruct the public key Q_B of node B. At the end of the bootstrapping phase, when B uses its private key q_B for the construction of the ECDH key K_{AB} and returns a MAC created with K_{AB}, the node A will have implicit assurance that it is talking to B and that all information included in the certificate is genuine (*i.e.* signed by the CA). Furthermore, a compromised node cannot present itself as a node of an earlier or a future generation. Each node's generation is included in its implicit certificate. If fake generation information j' is injected in IC_A for a corrupted node $A^{(j)}$, then the node B will construct an incorrect public key Q'_A and key confirmation will fail.

Usefulness of a network-wide authenticator. In both the hybrid and

fully asymmetric protocols, every node is pre-deployed with a network-wide authenticator K, in order to prevent trivial denial of service attacks from an outsider who does not know or possess any correct credentials. Let assume, for a moment, that the network-wide key K is not needed and that an outsider initiates the key agreement protocols (*e.g.*, playing the role of node B in Figure 3 and Figure 5). In the course of the attack, node A will detect the attack in Step 14 of the hybrid scheme and in Step 6 of the fully asymmetric scheme respectively, that is after performing the elliptic curve computations which are computationally intensive compared to any symmetric operation. This was the main reason why we adopted the solution of using the initial authenticator K. Admittedly however, this protection layer will not deter an adversary who captures any node of the network and learns the initial key K.

7 Performance evaluation

7.1 Computational complexity

In order to produce comparable results with related work, we use the metrics of [16] regarding the costs of each cryptographic action. Their computations were performed on Mitsubishi's 16-bit single-chip microprocessors M16C with 10MHz clock. The same metrics were used in the hybrid protocol of [18]. The costs per action are shown in Figure 6. The cost of fixed-point scalar multiplication is reduced, by having a pre-computed look-up table stored in the ROM area of each sensor. A block cipher, such as AES is assumed for the construction of the keyed-hash function. The keyed-hash function is used for the computation/verification of MACs. The SHA-1 algorithm is used for the evaluation of hash values, for random number generation and as the key derivation function kdf. The cost of the [16] protocol is 760 msec per node per key establishment. This is reduced to 645 msec for the hybrid protocol and to 633 msec for the asymmetric protocol, an improvement of 15-17%.

7.2 Communication complexity

The proposed protocols require a total of 4 message exchanges for each key establishment, including the protocol initiation, exchange of MACs and key confirmation. Assuming that the node ID is 64 bits, the generation ID and the expiration time are 8 bits, the Elliptic Curve modulus is 160 bits, the cipher-blocks and MACs are 128 bits, and the random nonces are 64 bits, then the communication cost of the protocol hybrid protocol is 186 bytes, while the cost of the asymmetric protocol is 180 bytes, equivalent with the cost of the [16] protocol.

7.3 Key storage requirements

Assuming that the nodes are pre-deployed with keys that allow communication with nodes of k generations (including their own generation), the total storage

Cryptographic Action	Cost/Action (msec)	Number of Actions Per Node			
		Hybrid Protocol		Asymmetric Protocol	
		Node A	Node B	Node A	Node B
Scalar multiplication (random point)	480	1	1	1	1
Scalar multiplication (fixed point)	130	1	1	1	1
EC addition	3	1	1	1	1
Symmetric encryption / decryption	3	2	2	-	-
Keyed hash function	3	5	6	4	4
Hash function evaluation	2	2	2	2	2
Random number generation	2	1	2	1	1
Total cost per node		**640**	**645**	**631**	**631**

Figure 6: Computational costs per node

requirements during key pre-deployment are $1032 + (k \times 128)$ bits for the hybrid protocol and $384 + k \times 400$ bits for the asymmetric protocol.

	Key pre-deployment storage costs per node for k node generations (bits)					
Compared protocols	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 10$
The hybrid protocol] $1032 + (k \times 128)$ bits	1160	1288	1416	1544	1672	2312
The asymmetric protocol $448 + (k \times 456)$ bits	904	1360	1816	2272	2728	5008

Table 1: Storage per node for k generations

As shown in table 1 the key pre-deployment storage costs allow for a reasonable number k of node generations. Note that although for $k \leq 2$ both protocols have almost equal storage costs, for $k > 2$ the hybrid protocols scales better. For $k = 10$ node generations, the asymmetric protocol has almost double key pre-deployment storage costs. However, the storage requirements of both protocols are tolerable for applications requiring a limited number of node generations. For example, for $k = 5$ node generations, the key storage costs are 2384 bits or

298 bytes.

This cost includes,the private EC keys of the sensor, the public key of the CA, the base point P and the symmetric keys. Note that after each key establishment phase, the sensor node will delete the EC keying material used in this phase. This helps in maintaining an almost constant key space regardless of the generation of pairwise keys.

8 Simulation results

In order to measure the efficiency of the proposed key establishment protocols under various node configurations, we have used the wireless libraries of the ns-2 (Network Simulator 2) [26]. For comparison reasons, we also simulate the hybrid protocol of Huang *et al* [16] under the same configurations, since this is the only existing hybrid protocol in the current literature for WSNs. In all cases examined, we have used the following assumptions:

At the physical layer, wireless communication between the nodes is assumed, as implemented in the ns-2 distribution. In the MAC layer, the IEEE 802.11 protocol is used, following the ns-2 distribution implementation. Routing is performed by the DSDV (Destination Sequenced Distance Vector) routing protocol. Finally, the UDP protocol is used as the transport agent.

8.1 Simulation setup and results

The simulated network consists of static nodes placed on a regular grid arrangement, with $30m$ vertices. The expected number of neighbors for each node is varying with each experiment, although a maximum number of 8 neighbors is set for every node. The propagation range is $50m$ by using the TwoRayGround model.

The key exchange protocols are coded as custom applications on the top of the ns-2 layered structure. This approach enables the accurate and realistic simulation of the key-exchange handshake mechanism. The application code functionality is as close as possible to a real implementation, with key establishment messages queued in finite buffers. The usage of queues permits the partial overlapping of sessions. This means that while waiting for a particular response, a node can process another key-exchange request. In order to speed up the ns-2 simulation, the actual cryptographic computations are not performed. Instead, we have pre-computed the delays of the uninterruptible work blocks of the protocols, using the computational costs presented in [16]. These delays are used for keeping busy a simulated node for the exact amount of time its handshake state dictates. During this period no other operation is possible, as in a real-life situation.

The simulation scenarios we have used can be divided into two categories: according to the first category, one or more key-server nodes are intermixed with ordinary nodes. The latter try to establish a key with one key-server in reach. In the second group of scenarios, one or more nodes exchange key pairs

with all neighbors in reach. All scenario cases, accompanied with the relevant simulation results are presented in the following sections.

8.2 Key-server scenarios

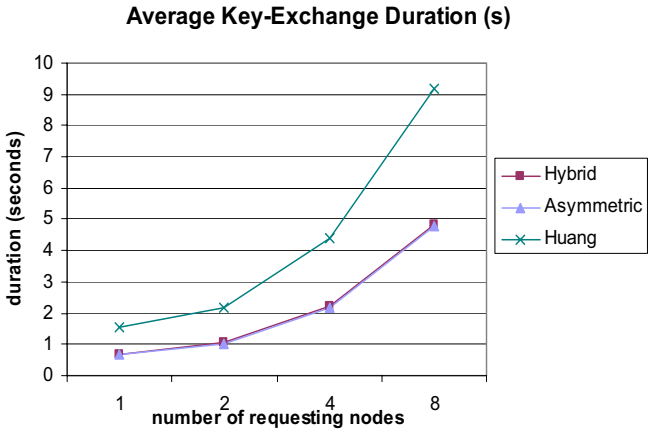

Figure 7: Average Key Exchange Duration

In the first scenario in this category, a key-server node is located between a varying number of requesting nodes (1 to 8). These nodes exchange keys with the server concurrently, generating overlapping key-establishment sessions on the server node. As shown in Figure 7, the duration of key-establishment sessions increases as the number of competing requestor nodes increases. The Asymmetric and Hybrid protocol graphs are nearly identical (only 1% in difference), but the Huang protocol, which is computationally more expensive and exchanges more messages, is almost 100% slower in every case. In the second scenario of the key-server category, a regular grid of 100 nodes is simulated for 3 different cases (A, B, C in Figure 8). In each case, a number of key-servers are interspersed within requesting nodes in a manner that allows for accordingly 1, 2 or 4 possible selections of a key-server. The first response to a key request establishes the requestor/key-server pair and all other server offers are rejected by the requesting node. As a key-server is busy with the first request it receives, without responding to other request for at least $485ms$ (Huang *et al* protocol), the distribution of key-server selection by requestors is implicitly load-balanced.

On the other hand, the work of a key-server in the same period ($654ms$ in Hybrid, $625ms$ in Asymmetric and $485ms$ in Huang *et al*) is totally wasted when a requestor selects another key-server to handshake. This is obvious from the graph in Figure 8, where the speedup between the case A and the cases B and C is always less than 18% (Hybrid, Asymmetric) or 36% (Huang *et al*), while at the same time, key-servers in cases B and C perform from 50% to 75% less exchanges than in case A. From this point of view, the Huang *et al* protocol has an advantage over the other protocols.

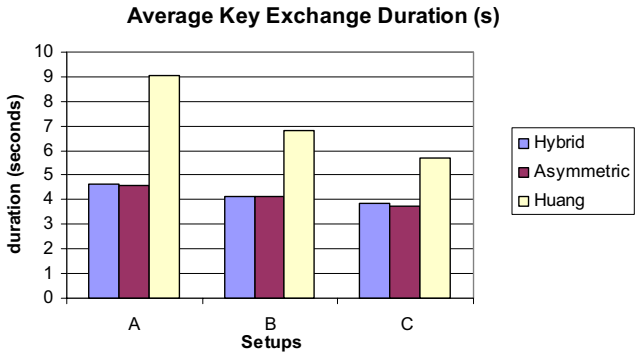

Figure 8: Average Key Exchange Duration

8.3 Pairwise exchange scenarios

In the simplest scenario of this category, a fresh node exchanges keys with all its neighbors. The number of neighbors varies between 1 to 8, causing overlapped and competing key-exchange sessions on requesting node.

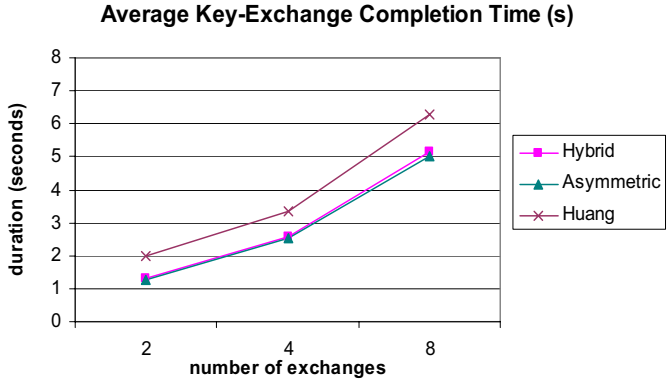

Figure 9: Average Key Exchange Completion Times

In Figure 9 the total key-exchange completion time is shown. Completion time increases as the number of concurrent sessions increases, but due to the possibility of overlapping the effect is less pronounced, especially in the case of the Huang protocol. According to the next scenario, a number of nodes exchange keys concurrently, each one with all its neighbors. Nodes are arranged in square grids and can have from 1 to maximum 8 neighbors. As the number of neighbors remains constant for the larger grids, this fact is reflected to the key-exchange completion times in Figure 10 after the 16 node (4x4) setup.

The slight increase in completion time observed in 32 and 64 node setups is caused by the delay of message routing in the system. As in previous con-

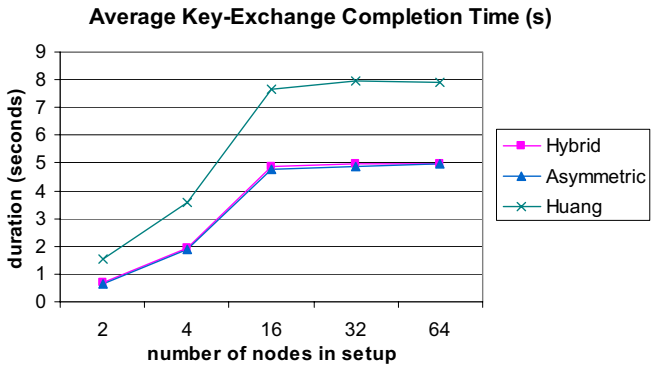

Figure 10: Average Key Exchange Completion Times

figurations, the Hybrid and Asymmetric protocols are 1.6 to 2.5 times faster than the Huang *et al* protocol, a difference however that tends to be lower when key-exchange sessions can be overlapped on a node.

9 Conclusion

In this paper we examine existing key establishment protocols for sensor networks and propose design and efficiency goals for Multi-Phase Distributed Sensor Networks (MPDSNs), where groups of sensors (*i.e.* generations) can join the network in future time periods and establish security relations with their team-mates and/or with older nodes of the network. We also propose a generic scheme for asymmetric key establishment in MPDSNs that fulfills our design goals. Finally, we examine, in both security and efficiency terms, two key establishment protocols for a MPDSN that make use of asymmetric cryptography, a hybrid and a fully asymmetric protocol [18, 23].

The hybrid protocol combines Elliptic Curve Diffie-Hellmann and Implicit Certificates with pre-deployed symmetric keys. The fully asymmetric protocol is a modified version of the hybrid, where no symmetric keys are pre-deployed to the nodes. Instead, each node is pre-deployed with multiple EC key pairs, one for each of the expected node generations.

To eliminate the costs of key generation for the sensor nodes, we use an off-line trusted Certification Authority CA, which is responsible to generate and pre-deploy the keys in a secure way. The authenticity of the EC keys of sensor nodes is based on Implicit Certificates, which are signed by the CA with EC-Schnorr signatures. Knowledge of the secret EC keys is implicitly confirmed during the key establishment.

The protocols are more secure to known-key security attacks than symmetric-key based protocols such as [35, 12], since they do not assume protection of the nodes during the key bootstrapping periods and are less vulnerable to several known weaknesses of symmetric authentication schemes. The proposed proto-

cols also improve the security of the existing hybrid protocol of [16], since they support multiphase node deployment and do not require the existence of fully functional devices.

The simulation results of the proposed protocols show that they have almost equal key exchange costs. Moreover, they have better performance in comparison with the existing hybrid protocol of [16], both in key-server and pairwise key exchange scenarios. Our results indicate that although the proposed protocols are still heavier than symmetric ones, their costs are affordable for highly sensitive applications that require advanced security protection.

References

[1] S. Basagni, K. Herrin, E. Rosti, and D. Bruschi, *Secure pebblenets*, Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM Press, 2001, pp. 156–163.

[2] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, *Perfectly secure key distribution for dynamic conferences*, Proceedings of the Advances in Cryptology - Crypto 92, Lecture Notes in Computer Science (LNCS), vol. 740, Springer-Verlag, 1992, pp. 471–486.

[3] D. Brown, R. Gallant, and S. Vanstone, *Provably secure implicit certificate protocols*, Proceedings of the 5th International Conference on Financial Cryptography, LNCS, vol. 2339, Springer-Verlag, 2002, pp. 156–165.

[4] S. Camtepe and B. Yener, *Key distribution mechanisms for wireless sensor networks:a survey*, Tech. Report TR-05-07, Rensselaer Polytechnic Institute, 2005.

[5] D. Carman, P. Kruus, and B. Matt, *Constraints and approaches for distributed sensor network security*, Tech. Report TR 00-010, NAI Laboratories, 2000.

[6] H. Chan and A. Perrig, *Security and privacy in sensor networks*, IEEE Computer **36** (2003), no. 10, 103–105.

[7] H. Chan, A. Perrig, and D. Song, *Random key predistribution protocols for sensor networks*, Proceedings of the IEEE Symposium on security and privacy (Berkeley, California), IEEE Press, May 2003, pp. 197–213.

[8] Crossbow, *Mica2 wireless measurement system datasheet*, Available at: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf, 2005.

[9] D. Dolev and A.C. Yao, *On the security of public key protocols*, IEEE Trans. Inf. Theory **29** (1983), 198–208.

[10] J. R. Douceur, *The sybil attack*, Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS 02) (Cambridge, MA (USA)), March 2002, pp. 251–260.

[11] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, *A key management scheme for wireless sensor networks using deployment knowledge*, Proceedings of the IEEE INFOCOM 04 (Los Alamitos, CA), March 2004.

[12] B. Dutertre, S. Cheung, and J. Levy, *Lightweight key management in wireless sensor networks by leveraging initial trust*, Tech. Report SRI-SDL-04-02, SDL, 2004.

[13] L. Eschenauer and V. D. Gligor, *A key-management protocol for distributed sensor networks*, Proceedings of the 9th ACM Conference on Computer and Communications Security, ACM Press, 2002, pp. 41–47.

[14] G. Gaubatz, J. P. Kaps, and B. Sunar, *Public key cryptography in sensor networks -revisited*, Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS '04), 2004.

[15] C. Gunther, *An identity-based key-exchange protocol*, Proceedings of the Advances in Cryptology – Eurocrypt '89, Lecture Notes in Computer Science (LNCS), vol. 434, Springer-Verlag, 1990, pp. 29–37.

[16] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, *Fast authenticated key establishment protocols for self-organizing sensor networks*, Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, ACM Press, 2003, pp. 141–150.

[17] K. Jamshaid and L. Schwiebert, SEKEN *(secure and efficient key exchange for sensor networks)*, Proceedings of the 23rd IEEE International Performance, Computing, and Communications Conference (IPCCC), 2004.

[18] P. Kotzanikolaou, E. Magkos, C. Douligeris, and V. Chrissikopoulos, *Hybrid key establishment for multiphase self-organized sensor networks*, Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks WoWMoM '05, IEEE Press, 2005, pp. 581–587.

[19] H. Krawczyk, *Hmqv: A high-performance secure diffe-hellman protocol*, Proceedings of the Advances in Cryptology - Crypto 2005, Lecture Notes in Computer Science (LNCS), vol. 3621, Springer-Verlag, 2005, pp. 546–566.

[20] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, *An efficient protocol for authenticated key agreement*, Designs, Codes and Cryptography **28** (2003), no. 2, 119–134.

[21] L. Law, A. Menezes, M. Qu, J. Solinas, and S.A. Vanstone, *An efficient protocol for authenticated key agreement*, Technical Report CORR 98-05, Department of C & O, University of Waterloo, 1998.

[22] D. Liu, P. Ning, and R. Li, *Establishing pairwise keys in distributed sensor networks*, ACM Transactions on Information and System Security **8** (2005), no. 1, 41–77.

[23] E. Magkos, P. Kotzanikolaou, M. Stefanidakis, and D. Vergados, *An asymmetric key establishment protocol for multiphase self-organized sensor networks*, Proceedings of the 12th European Wireless Conference (EW '06), March 2006.

[24] D. Malan, M. Welsh, and M. Smith, *A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography*, Proceedings of the 1st IEEE International Conference on Sensor and Ad hoc Communications and Networks (Santa Clara, California), IEEE Press, October 2004.

[25] A. Menezes, M. Qu, and S. Vanstone, *Key agreement and the need for authentication*, PKS '95 (Toronto, Canada), November 1995.

[26] NS-2, *The network simulator ns-2,* C*ornell* U*niversity*, Available at: http://www.isi.edu/nsnam/ns/, 2005.

[27] L.B. Oliveira, H. C. Wong, and A. A. F. Loureiro, LHA-SP*: Secure protocols for hierarchical wireless sensor networks*, Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management (IM'05), 2005, pp. 31–44.

[28] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D. E. Culler, SPINS*: security protocols for sensor networks*, Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01), July 2001, pp. 189–199.

[29] S. Schmidt, H. Krahn, S. Fischer, and D. Watjen, *A security architecture for mobile wireless sensor networks*, Proceedings of the 1st European Workshop on Security in Ad-Hoc andSensor Networks (ESAS '04) (Heidelberg, Germany), vol. 3313, August 2004.

[30] C. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology **4** (1991), 161–174.

[31] SECG, *Standards for efficient cryptography group.* SEC *1: Elliptic curve cryptography*, Available at: http://www.secg.org/download/aid-385/sec1_final.pdf, 2005.

[32] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck, and M. B. Srivastava, *On communication security in wireless ad-hoc sensor networks*, Master's thesis, Proceedings of the 11th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02), 2002, pp. 139–144.

[33] R. Struik and G. Rasor, *Mandatory ecc security algorithm suite*, Available at: http://grouper.ieee.org/groups/802/15/pub/2002/May02/02200r1P802-15_TG3-Mandatory-ECC-Security-Algorithm-Suite.pdf, 2002.

[34] A.D. Wood and J.A. Stankovic, *Denial of service in sensor networks*, IEEE Computers **35** (2002), 54–62.

[35] S. Zhu, S. Setia, and S. Jajodia, *Leap: efficient security mechanisms for large-scale distributed sensor networks*, Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03) (Washington D.C.), October 2003, pp. 62–72.