Hybrid Key Establishment for Multiphase Self-Organized Sensor Networks

Panayiotis Kotzanikolaou
University of Piraeus
Department of Informatics
80 Karaoli & Dimitriou, Piraeus 185 34, Greece
pkotzani@unipi.gr

Christos Douligeris
University of Piraeus
Department of Informatics
80 Karaoli & Dimitriou, Piraeus 185 34, Greece
cdoulig@unipi.gr

Emmanouil Magkos
Department of Archiving and Library Studies,
Ionian University, Old Palace
Corfu, 49100, Greece
emagos@ionio.gr

Vassilios Chrissikopoulos
Department of Archiving and Library Studies,
Ionian University, Old Palace
Corfu, 49100, Greece
vchris@ionio.gr

Abstract

Recent work on key establishment for sensor networks has shown that it is feasible to employ limited Elliptic Curve Cryptography in sensor networks through hybrid protocols. In this paper, we propose a hybrid key establishment protocol for uniform self-organized sensor networks. The proposed protocol combines the Elliptic Curve Diffie-Hellmann key establishment with implicit certificates and symmetric-key cryptographic techniques. The protocol can be implemented on uniform networks comprised of restricted functional devices. Furthermore, due to its public-key nature, the protocol is resilient to a wide range of passive and active attacks such as known-key attacks, as well as attacks against the confidentiality, integrity and authenticity of the communication. The protocol is scalable and efficient for low-capability 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.

1 Introduction

Distributed Sensor Networks (DSN) are becoming increasingly popular as they can be used in a variety of civil applications [9]. Of special interest is the usage of DSNs in secure-critical domains: thousands of such nodes could be deployed in unattended and/or adversarial environments to collect information such as tracking hostile troop movements, or detecting chemical and biological weapons.

Wireless networks are in general more vulnerable to security threats than wired networks [4]. Depending on the environment where nodes are deployed, appropriate protection measures should be taken for data confidentiality, integrity and authentication between communicating entities, while taking into account the cost, storage, energy and communication efficiency requirements. To support such security services one needs key management techniques that are resilient to both external and internal attacks. The required trade-off, makes it an important challenge to design secure and efficient key establishment techniques for DSNs.

An on-going research area is establishing secure communication channels between pairs of sensor nodes, especially in DSNs that do not rely on any fixed infrastructure or central administrator. This infrastructureless model is known as the *self-organizing* model [6]. The self-organizing model can be further divided into two cases. In the *non-uniform* self-organizing model the network may contain two types of nodes; the full functional devices (FFD) with high energy, power and storage capabilities, and the restricted functional devices (RFD), which are the typical low-capability sensor nodes. In the *uniform* self-organizing model, all the nodes are assumed to be restricted functional devices. Obviously, achieving the required security level is more difficult in the uniform model, since one can only rely on low-capability devices. In this paper, we are concerned with key management in uniform self-organizing DSNs.

Multiphase deployment. In highly dynamic and decentralized environments, it may be required that the network is updated with other nodes in future time periods, in order to extend the network or replace erroneous nodes. Each set of incoming nodes that will join the network in a future time

consist a *node generation*. The nodes of a particular generation are pre-deployed with the appropriate keys, which will enable them to perform key bootstrapping with each other, as well as with nodes of a previous generation. The protocols which allow multiple key bootstrapping phases between nodes of different generations are known as *multiphase* deployment protocols [5].

Due to the constrained environment, symmetric cryptography seems to be the most efficient choice for multiphase key establishment in DSNs. In [5, 14], two symmetric-key approaches were proposed. In these schemes all nodes of a certain node generation i are pre-deployed with a generation-wide symmetric key K_i, which they will use during the i_{th} pairwise key bootstrapping period. Each node A belonging to the i_{th} generation is also pre-deployed with a different "instance" of the future generation keys, linked to its unique identity ID_A, *i.e.* $K_{i+1}(ID_A) = f_{K_{i+1}}[ID_A]$, ..., $K_m(ID_A) = f_{K_m}[ID_A]$, where f is a one-way keyed hash function, and m is the total number of generations. Each of these keys will be used by the node to participate in the corresponding future bootstrapping phase. At the end of each bootstrapping period, all nodes delete their generation key (or their instance of the generation key respectively).

In the protocols of [5, 14], it is assumed that the nodes cannot be attacked during the bootstrapping period. This is a strong assumption, since the sensors cannot be tamper-resistant due to their physical limitations. Moreover, if a node is compromised during the bootstrapping period, then the corrupted node may use the generation-wide key to impersonate any other node, an attack also known as *sybil attack*. Finally, since only symmetric encryption techniques are employed, the nodes cannot prove their participation in a specific node generation. This could be useful in some circumstances, *e.g.* when a fresh node must be programmed to cooperate with sensors of a specific generation, or when inter-generation communication shall be given higher priority. In such cases, the above protocols may be subject to *fake generation* attacks: corrupted nodes could pretend to belong to another node generation than the authentic one.

Public-key cryptography for DSNs. Although the costs of public key encryption are prohibitive for sensor nodes, recent research has shown that it is possible to construct sensors capable of performing (limited) public key cryptography [7], mainly through Elliptic Curve (EC) cryptography [3] – see [11, 8]. Huang *et al* [8] proposed a hybrid protocol for pairwise key establishment in DSNs, by combining EC and symmetric techniques. To minimize the number of the expensive scalar multiplications, the authors in [8] propose the employment of some full-functional devices that take most of the cryptographic burden. The cost for each restricted sensor node is then reduced to one scalar multiplication with a random point and one with a static point, per key establishment. This cost is tolerable for security-critical DSNs. Although the use of EC cryptography may overcome the security issues of symmetric techniques for multiphase deployment, the scheme of [8] is not appropriate for uniform self-organized DSNs, since it assumes the existence of full-functional devices. Moreover, this protocol is designed for single phase node deployment.

Our contribution. We propose a hybrid key establishment protocol, for multiphase deployment in uniform self-organizing sensor networks. The protocol combines standard Elliptic Curve Diffie-Hellmann (ECDH) [3] with symmetric encryption techniques. The authentication of the EC keys is based on Implicit Certificates [13], issued by an off-line Certification Authority. The computation cost of the EC cryptographic actions for each sensor is reduced to a scalar multiplication over a static point and a scalar multiplication over a random point. The cost reduction is due to the combination of symmetric encryption in the randomization process and the use of EC-Schnorr signatures [12] in the Implicit Certificate verification. The protocol is scalable, with sensors being pre-deployed with a constant number and size of keys, regardless of the size of the network.

The proposed protocol improves over the symmetric-key based schemes of [5, 14], as it does not allow a compromised node to impersonate other nodes, belonging to the same or a different generation. Furthermore, it provides forward secrecy both in respect to a particular node and a generation of nodes. Moreover, it does 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 protocol improves over the hybrid scheme of [8], since it supports multiphase deployment, and does not require the existence of full-functional devices.

2 A Hybrid Key Establishment Protocol for Self-Organized DSNs

We propose a key establishment protocol for sensor networks, which combines EC with symmetric cryptographic techniques to support secure multiphase deployment in the uniform self-organizing model. Before initialization of the network a trusted authority CA pre-deploys each sensor with the appropriate EC and symmetric keys. After the key pre-deployment the CA stays off-line and it is no further involved in the protocol. The nodes are randomly deployed (*e.g.* via aerial scattering) and are not aware of their neighbors until their deployment. In the sequel, the nodes participate in a bootstrapping phase in order to exchange keys with their neighboring nodes. After the initialization of the network, it is possible for sensor nodes to join the network in future time periods, provided they have been pre-deployed (by the CA) with the corresponding generation keys.

2.1 Notation

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}.

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 [14]; however, in our protocol these keys will only be used to create a temporary channel for exchanging randomness for the key establishment, to mitigate the consequences of a static key being compromised and to establish forward secrecy for exchanged session keys.

2.2 Key pre-deployment phase

The CA generates and pre-deploys each node with the appropriate keying information (see Figure 1). We describe the key generation and key pre-deployment 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 [3]. Then, the CA computes the static secret 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 [12], created by the CA, over the message M of the Implicit Certificate IC_X of the node X. The corresponding public 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.

[1] Informally, the idea is simply to view the octet string as the base 256 representation of the integer (Section 2.3.8 of [3])

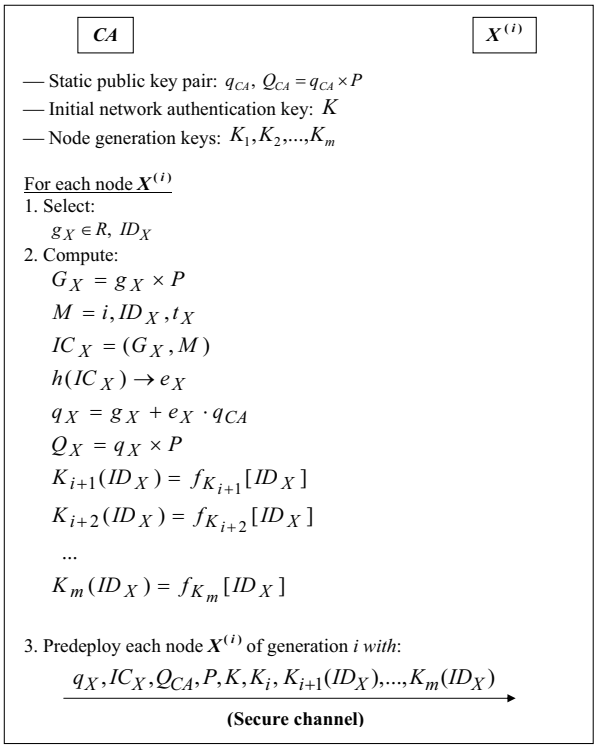

Figure 1. The key pre-deployment phase

After 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 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 key of the i_{th} generation K_i and the instance keys $K_{i+1}(ID_X), K_{i+2}(ID_X), ..., K_m(ID_X)$.

The role of the CA is different from the traditional PKI model. The CA acts as a trusted authority that generates and pre-deploys off-line the appropriate keys to each node. The nodes do not compute or verify the validity of their keys. This eliminates the communication and computation costs for the sensor nodes, during the key pre-deployment phase. After the initialization of the network the CA will have a passive role, and will not further participate in key establishment. The CA will only be allowed to generate and pre-deploy the keys for nodes of forthcoming generations.

2.3 Key establishment phase

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 2).

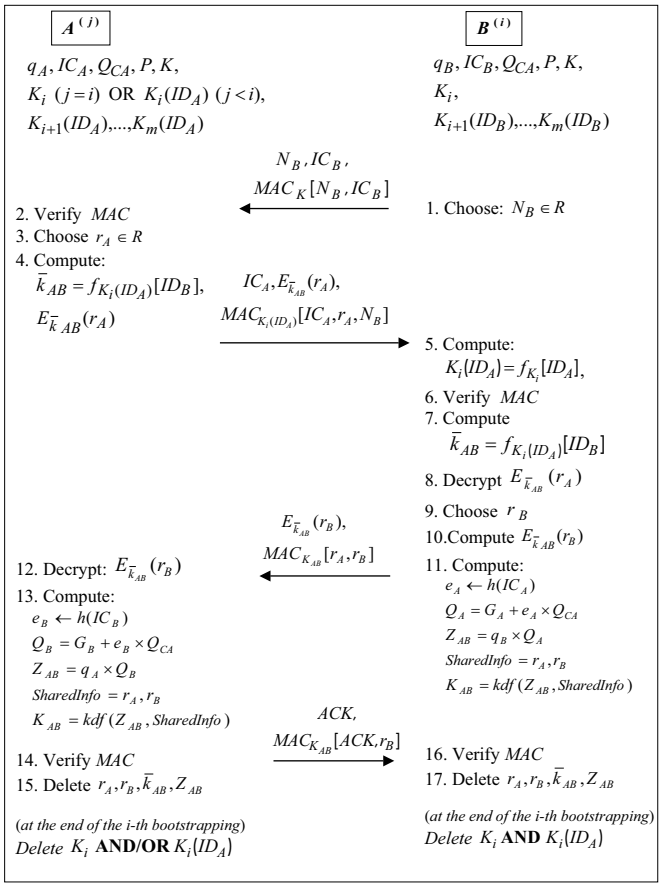

Figure 2. Key bootstrapping phase

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.

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 authentication key K. The neighboring node A receives and verifies the MAC and if the verification succeeds, it chooses a random number r_A, which will be used in the randomization of the ECDH key exchange.

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 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 $K_i(ID_A)$.

On receiving this message, the node B computes the key $K_i(ID_A)$, by using its generation key K_i. Then, B checks the received MAC and if it verifies correctly, the node B computes the temporary key $\bar{k}_{AB}$ by using the already computed key $K_i(ID_A)$. The node B will then decrypt $E_{\bar{k}_{AB}}[r_A]$ and obtain r_A. 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 $\bar{k}_{AB}$.

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* [13] 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.

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 $Q_B = G_B + e_B \times Q_{CA}$. At this point node A is not assured about the authenticity of Q_B. The authenticity of

[2] In standard $ECDH$ [3], $SharedInfo$ is an optional string including some mutually known private information (specified as $suppPrivInfo$).

this key will be assured only after node A establishes B's knowledge of the corresponding secret key q_B.

The node 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 $SharedInfo = r_A, r_B$. 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}.

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. After this 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 [3]. For key freshness, the nodes can periodically update the pairwise key with an one-way hash function. The time interval between subsequent renewals may depend on the data traffic volume, as well as on the strength of the underlying cryptographic primitives.

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 $i + 1$.

3 Security Analysis

The proposed protocol extends standard ECDH key agreement [3], by randomizing the key generation in order to protect from known-key security attacks. Our protocol is hybrid. Its "symmetric" part is a four-pass challenge-response variation of AKEP2 [1] to support authenticated key agreement with explicit key confirmation using initial trust between sensors nodes. During bootstrapping sensor nodes use this initial trust to exchange randomness, which will be used together with the static ECDH key, as an input to a key derivation function. The "symmetric" part of the proposed protocol can be proven secure by using the models and techniques described in [1]. We 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.

The "public" part of the protocol involves the standard ECDH key agreement [3] combined with implicit certificates for mutual authentication. The authenticity of the implicit certificates is based on EC-Schnorr [12] signatures, which are provably secure under the *random oracle* model given that the discrete logarithm problem over a subgroup $< G >$ is untractable [2].

3.1 Known-key security

By using a private off-line interface between each sensor node and the CA, during the pre-deployment phase, both active and passive attacks against the key generation process (such as unknown key share attacks and small subgroup attacks [10]) are thwarted, provided that the CA is honest and takes all reasonable measures in the key generation process.

In the following, we examine the security of the key establishment and consequently the secrecy and integrity of the exchanged communication between two nodes against key compromisation by an active adversary who also eavesdrops on communication channels. Clearly, if both *a*) the generation key K_i (or the instance $K_i(ID_A)$ used in the temporary key generation) and *b*) the secret key q_A of node A are compromised, then the generated pairwise key K_{AB} is compromised. Moreover, if the adversary has recorded all previous communication of node A, then the adversary will be able to compute all former pairwise keys of that node. We will examine the security of the exchanged keys in the cases where either the generation key or the static secret key of a node is compromised.

Security against compromisation of the generation key. If a generation key K_i, is compromised, then all the temporary keys $\bar{k}_{AB} = f_{K_i(ID_A)}[ID_B]$ between any pair of nodes A and B will be compromised, and consequently the random values r_A and r_B will be revealed. However, the pairwise key also relies on the static ECDH key Z_{AB}. Note that the static pairwise key Z_{AB} is generated only once during the key establishment phase between the two nodes and after the key establishment Z_{AB} is deleted. The key Z_{AB} is never used for encryption, thus even if the adversary has recorded previous communication, cannot obtain ciphertexts with that key to cryptanalyse and obtain the static ECDH key. The best the adversary can do is to attempt to obtain the key $K_{AB} = kdf(Z, r_A, r_B)$ for random values Z. Furthermore, the node A participates in key establishment with any other node only once. Thus the adversary will not be able to cryptanalyse ciphertexts produced with keys of the form $K_{AB} = kdf(Z_{AB}, r_A, r_B)$, $K'_{AB} = kdf(Z_{AB}, r'_A, r'_B)$, ..., for known random values with the same static key Z_{AB} and in this way obtain Z_{AB}. Consequently, the adversary will not be able to obtain the pairwise key K_{AB}, provided that the length of the static key Z_{AB} is sufficiently large and that the static secret EC keys of the two nodes A and B have not been compromised. This implies that our protocol can resist compromisation of the generation-wide key, since this will not automatically compromise the keys established

by nodes of this generation. Of course, by preserving the secrecy of the generation-wide key (as assumed in [5, 14]), the security of the protocol is further increased, since one would also have to obtain the random values exchanged by the nodes in the key establishment.

Security against compromisation of the static secret key. If only the static key q_A of a node A is compromised, while the generation key K_i is secure, then the adversary will be able to obtain any static pairwise key of that node, say Z_{AB}. However, the adversary will not be able to obtain the random values r_A, r_B used in the key derivation process, provided that the random values are sufficiently large. Thus the pairwise key K_{AB} cannot be obtained, provided that the random nonces r_A and r_B are sufficiently large to withstand cryptanalytic attacks against $K_{AB} = kdf(Z_{AB}, r_A, r_B)$.

Forward secrecy (per node). Stealing the keys of a node at a given time does not reveal past communications of the attacked node. Indeed, 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. Thus, even if the static ECDH key is revealed, the past session keys cannot be computed without knowledge of the particular random values used for each past session key. Recall that at the end of the 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 are used to exchange r_A and r_B. Thus all past communications of the compromised node are secure.

Forward and backward secrecy (per generation). Stealing all the keys of a node (including the generation-wide key) does not compromise past or future communications of any other node of the same generation. Computation of a session key for each pair of nodes relies on both the generation-wide key and the static secret key of each particular node. Thus, the compromization of the generation-wide key will not reveal past of future communication keys of any other node belonging to the same node generation. Obviously, this also holds for nodes belonging to different node generations.

3.2 Node authentication

Security against impersonation attacks. To prevent impersonation attacks we make use of implicit certificates [13]. During the 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).

Security against fake generation attacks. A compromised node cannot present itself as a node of an earlier or 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 in step 11 (see figure 2) node B will construct an incorrect public key Q'_A and key confirmation will fail in step 16.

4 Performance Evaluation

We evaluate the performance of the proposed protocol in terms of computation and communication cost, as well as the storage requirements regarding the pre-deployed EC and symmetric keys required.

Computational complexity. In order to produce comparable results with related work, we use the metrics of [8] regarding the costs of each cryptographic action. Their computations were performed on Mitsubishi's 16-bit single-chip microprocessors M16C with 10MHz clock. The costs per action are shown in Figure 3. For symmetric encryption/decryption the AES block cipher in CBC mode is assumed, for text blocks of 256-bit length. AES can also be used for the construction of the keyed-hash function. The keyed-hash function is used both for the computation/verification of MACs, as well as for the computation of the symmetric keys (i.e. the function f used in the protocol.) The SHA-1 algorithm is used for the evaluation of hash values, for the generation of random values and as the key derivation function kdf. The computation evaluation of our protocol shows a total cost per node of about 645 msec. This cost is about 20% lower than the cost of the hybrid protocol of [8] (760 msec) computed with the same metrics.

Cryptographic Action	**Cost/action (msec)**	**Number of actions per node**	
		Node A	**Node B**
Scalar multiplication (random point)	480	1	1
Scalar multiplication (fixed point)	130	1	1
EC addition	3	1	1
Symmetric Encryption / Decryption	3	2	2
Key Hash Function evaluation	3	5	6
Hash Function Evaluation	2	2	2
Random number evaluation	2	1	2
Total Computation Cost per node		640	645

Figure 3. Computational Costs per Node

Communication complexity. The proposed protocol requires a total of 4 message exchanges for key establishment, including the protocol initiation, exchange of randomness, exchange of MACs and key confirmation. Assuming a node ID is 64 bits, a generation ID is 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 is 1488 bits or 186 bytes, almost equal to the 180 bytes required in [8].

Storage requirements. Assuming that the nodes are pre-deployed with keys that allow communication with nodes of k generations (including their own generation), then the total storage requirements during key pre-deployment are $1032 + k * 128$ bits. For example, for a network of five node generations, each node is pre-deployed with 1544 bits or 193 bytes. Our protocol is scalable: for a fixed number of generations, the per-node storage and energy resources do not limit the size of the network.

5 Concluding Remarks

In this paper we propose a hybrid key establishment protocol, suitable for uniform self-organized sensor networks. The proposed protocol is based on the standard ECDH key establishment protocol. In order to minimize the cost of scalar multiplications, the protocol uses a temporary encrypted channel to protect the randomization of the established pairwise keys. 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 and certificates 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 protocol is more secure to known-key security attacks than symmetric-key based protocols [14, 5] since it does not assume protection of the nodes during the key bootstrapping periods. The protocol provides forward secrecy per node and per generation. Moreover, corrupted or captured nodes cannot perform impersonation, sybil or fake generation attacks to any node, other than the corrupted one. The proposed protocol improves on the hybrid protocol of [8], since it supports multiphase node deployment and does not require the existence of full-functional devices. The computation, cost per node for a key establishment is slightly less than the protocol of [8], while the communication and storage costs are comparable.

References

[1] M. Bellare, and P. Rogaway. Entity Authentication and Key Distribution. Proc. of Crypto '93, LNCS Vol. 773, Springer-Verlag, pp. 232-249, 1994.

[2] D. R. L. Brown, R. P. Gallant, and S. A. Vanstone. Provably Secure Implicit Certificate Schemes. Proc. of FC'02, LNCS, Vol. 2339, pp. 156–165, 2002.

[3] Certicom Research, Standard for efficient cryptography, SEC 1: EC Cryptography. Ver. 1.0, 2000.

[4] H. Chan and A. Perrig. Security and Privacy in Sensor Networks. In: IEEE Computer, Vol. 36(10), pp. 103-105, October 2003.

[5] B. Dutertre, S. Cheung, and J. Levy. Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust. TR SRI-SDL-04-02, April 2004.

[6] L. Eschenauer, and V. D. Gligor. A key-management scheme for distributed sensor networks. Proc. of 9th CCS ACM conference, pp.41-47, 2002.

[7] G. Gaubatz, J. P. Kaps, and B. Sunar. Public key cryptography in sensor networks –revisited. Proc. of 1st ESAS Conference, 2004.

[8] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. Proc. of 2nd ACM WSNA Conference, pp.141-150, 2003.

[9] J. M.Kahn, R. H. Katz, and K.S.J. Pister. Mobile Networking for Smart Dust. Proc. of Mobicom'99, ACM/IEEE, Seattle, WA, August 1999.

[10] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An Efficient Protocol for Authenticated Key Agreement. In: Designs, Codes and Cryptography, Vol. 28(2), pp. 119–134, March 1998.

[11] D. Malan, M. Welsh, and M. Smith. A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography. Proc. of 1st SACN Conference, IEEE, 2004. 2003.

[12] C. Schnorr. Efficient Signature Generation by Smart Cards. In: Journal of Cryptology, Vol. 4 (1991), pp. 161-174.

[13] R. Struik, and G. Rasor. Mandatory ECC Security Algorithm Suite. Submissions to IEEE p802.15 Wireless Personal Networks, April 2002.

[14] S. Zhu, S. Setia and S. Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. Proc. of CCS'03, ACM, October 2003.