Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 171 (2007) 17–31
www.elsevier.com/locate/entcs

Design, Analysis and Performance Evaluation of Group Key Establishment in Wireless Sensor Networks [1]

Ioannis Chatzigiannakis[a,2], Elisavet Konstantinou[b,3], Vasiliki Liagkou[a,4] and Paul Spirakis[a,5]

[a] *Research Academic Computer Technology Institute (CTI), and Department of Computer Engineering and Informatics, University of Patras, 26500 Rio, Greece*

[b] *Dep. of Information and Communication Systems Engineering University of the Aegean, 83200 Samos, Greece*

Abstract

Wireless sensor networks are comprised of a vast number of ultra-small autonomous computing, communication and sensing devices, with restricted energy and computing capabilities, that co-operate to accomplish a large sensing task. Such networks can be very useful in practice, e.g. in the local monitoring of ambient conditions and reporting them to a control center. In this paper we propose a new lightweight, distributed group key establishment protocol suitable for such energy constrained networks. Our approach basically trade-offs complex message exchanges by performing some amount of additional local computations. The extra computations are simple for the devices to implement and are evenly distributed across the participants of the network leading to good energy balance. We evaluate the performance our protocol in comparison to existing group key establishment protocols both in simulated and real environments. The intractability of all protocols is based on the Diffie-Hellman problem and we used its elliptic curve analog in our experiments. Our findings basically indicate the feasibility of implementing our protocol in real sensor network devices and highlight the advantages and disadvantages of each approach given the available technology and the corresponding efficiency (energy, time) criteria.

Keywords: Group Key Establishment, Wireless Sensor Networks, Distributed Algorithms, Algorithm Design, Performance Evaluation

[1] This work has been partially supported by the IST Programme of the European Union under contract number IST-2005-15964 (AEOLUS), the Programme PYTHAGORAS under the European Social Fund (ESF) and Operational Program for Educational and Vocational Training II (EPEAEK II)

[2] Email: ichatz@cti.gr

[3] Email: ekonstantinou@aegean.gr

[4] Email: liagkou@cti.gr

[5] Email: spirakis@cti.gr

doi:10.1016/j.entcs.2006.11.007

1 Introduction

Wireless Sensor Networks are considered as very large systems comprised of small sized, low-power, low-cost sensor devices that collect detailed information about the physical environment. Each device has one or more sensors (e.g. light, heat, movement, chemical presence, etc.), embedded processors and low-power radios, and is normally battery operated. Examining each such single device individually, might appear to have small utility. The realization of Sensor networks however, lies in using and co-coordinating a vast number of such devices and allows the implementation of very large sensing tasks. The system is deployed in areas of interest (ranging from homes to inaccessible terrains, disaster places, etc.) making them smart spaces where fine grained monitoring services and applications can be provided [1].

The unique characteristics of this regime give rise to very different design trade-offs than current general-purpose systems. The realization of such efficient, robust and secure ad-hoc networking environments is a challenging algorithmic, systems and technological task. Large numbers of such tiny and resource-constrained devices should self-organize into an ad-hoc network under highly dynamic ambient conditions, carrying out computations locally and engaging into a collaborative computing and communication effort. The required solutions differ significantly, not only with respect to classic distributed computing but also with respect to ad-hoc networking. To further emphasize on the difference consider that in sensor networks (i) the number of interacting devices is extremely large and dense compared to that in a typical ad-hoc network, (ii) the resources of each node are very limited, (iii) there is no fixed infrastructure, (iv) the network topology is unknown before deployment and (v) there is a high risk of physical attacks in unprotected sensor nodes.

Such systems should at least guarantee the confidentiality and integrity of the information reported to the controlling authorities regarding the realization of environmental events. Therefore, key distribution is critical for the protection in wireless sensor networks and the prevention of adversaries from attacking the network. However, key management and establishment can be a difficult task in such networks and may waste the limited energy resources of the devices. The constraints of sensor node hardware influence the type of security mechanisms and protocols that can be hosted on a sensor node platform. Moreover, the ad hoc networking topology makes it susceptible to link attacks ranging from passive eavesdropping to active interference. Therefore, the choice of a key establishment protocol for the creation of a shared, secret key must be done very carefully and should exhibit the following critical properties: (i) *availability*, in the sense that any sensor node or service of the whole wireless network must be available whenever required, (ii) *key authentication*, assuring only intended nodes can access a key, (iii) *integrity*, ensuring that there is no unauthorized data modification and (iv) *confidentiality*, by providing security measures in order to avoid eavesdropping. Furthermore, some additional requirements are needed for the evaluation of key distribution in wireless sensor networks: (a) *scalability*, in the sense that they should operate efficiently in extremely large

networks composed of huge numbers of nodes, (b) *efficiency* with respect to both energy and time and (c) *fault-tolerance* as sensor devices are prone to several types of faults and unavailabilities, and may become inoperative (permanently or temporarily). In this sense, group key establishment is potentially more suitable than pairwise key establishment as sensors do not waste energy every time they wish to communicate with another device by establishing a new shared secret key.

Group key management mainly includes activities for the establishment and the maintenance of a group key. Secure group communication requires scalable and efficient group membership with appropriate access control measures to protect data and to cope with potential compromises. A secret key for data encryption must be distributed with a secure and efficient way to all members of the group. Another important requirement of group key management protocols is key freshness. A key is fresh if it can be guaranteed to be new. Moreover, the shared group key must be known only to the members of the group. Four important cryptographic properties must be encountered in group key agreement [20,22]. Assume that a group key is changed m times and the sequence of successive keys is $\mathcal{K}=\{K_0,\ldots,K_m\}$.

(i) **Computational group key secrecy**: It guarantees that it is computational infeasible for any passive adversary to discover any group key $K_i \in \mathcal{K}$ for all i.

(ii) **Decisional group key secrecy**: It ensures that there is no information leakage other that public blinded key information.

(iii) **Key independence**: It guarantees that a passive adversary who knows a proper subset of group keys can not discover any other of the remaining keys. Key independence can be decomposed into **forward secrecy** and **backward secrecy**. Forward secrecy guarantees that a passive adversary who knows a contiguous subset of old group keys cannot discover any subsequent group key. Backward secrecy guarantees that a passive adversary who knows a contiguous subset of group keys cannot discover preceding group key.

Group key establishment can be either centralized or distributed. In the first case, a member of the group is responsible for the generation and the distribution of the key. In distributed group key establishment all group members contribute to the generation of the key. Clearly, the second approach is suited for sensor networks because problems with centralized trust and the existence of single point of failure can be avoided. In our paper, we consider distributed group key establishment protocols [5,10,14,28] which can be applied in dynamic groups (where members can be excluded or added) and provide forward and backward secrecy. Moreover, all these protocols are based on the Diffie-Hellman key exchange algorithm [13] and constitute natural extensions of it in the multiparty case.

Related Work and Comparison. Most group key establishment protocols are based on generalizations of Diffie-Hellman key exchange protocol [13]. The first attempt for the construction of such protocols was made by Ingemarsson, Tang and Wong [14] that arrange the participants in a logical ring via a synchronous start up phase. The protocol completes in $n-1$ rounds, where n is the number of the participants.

Burmester and Desmedt presented in [10] a more efficient scheme which requires only three rounds. However, the protocol's disadvantage is that (i) every participant must perform $n+1$ exponentiations and (ii) communication is based on concurrent broadcasts that lead to high number of collisions, a situation very common in wireless sensor networks that affects performance [12]. Moreover, the authors do not provide a proof of security (in the stronger sense of semantic security). Recently, Katz and Yung [19] proposed a more general framework that provides a formal proof of security for this protocol. In Hypercube protocol [5] the participants in the network are arranged in a logical hypercube. This topology decreases the number of transmitted data and exponentiation operations, but still the protocol is very demanding for use in sensor networks.

One of the most efficient protocols in the literature for group key management is the third protocol GDH.3 of Steiner, Tsudik and Waidner presented in [28]. This protocol requires serial execution of computations that makes it inefficient for highly dynamic networks with large number of nodes. More precisely, this protocol may not be a good choice for an dynamically evolving ad hoc environment since the last node in the protocols computation would have to know the whole structure of the network.

A performance analysis of all the above mentioned protocols is presented in [3,2] which clearly shows the superiority of GDH.3 protocol in the number of transmitted data and exponentiation operations required. They show that the number of messages and exponentiations is linear to the number of the participants, while for all other protocols are of order $n \log n$ or n^2.

A very efficient protocol is also presented in [20]. In this recent work, a logical key tree structure is used to improve the scalability of the key agreement protocol. Any device can calculate the group key if it knows all the keys in its co path. This requirement makes the protocol quite expensive in storage memory that is critical for sensor networks. For these reasons, we believe that the simplicity and the limited memory requirements of GDH.3 protocol make it more suitable and applicable in sensor networks. Finally, we note that the recent papers of Bresson et al. [7,8,9] were the first to present a formal model of security for group authenticated key exchange and the first to give rigorous proofs of security for particular protocols.

Main Findings. In this work, we propose a new distributed group key management protocol suitable for such energy constrained networks. This protocol resembles GDH.3 in its first stage but instead of requiring direct communication among the participants of the network, it relies on short-range hop-by-hop propagation. To do this we employ a larger number of public key encryption and decryption operations per sensor node. The extra computations are simple for the devices to implement and are evenly distributed across the participants of the network, leading to fewer number of message exchanges. Our group key management protocol handles membership events like join or leave in order to provide forward and backward secrecy.

We implementated our protocol using the elliptic curve version of Diffie-Hellman problem [13]. This allows us to use much smaller keys than conventional, discrete

logarithm based cryptosystems (an 160-bit key in an elliptic curve cryptosystem provides equivalent security with a 1024-bit key in a conventional cryptosystem). This fact makes elliptic curves the only reasonable choice for sensor networks, where the resources are very limited. Moreover, recent research has shown that public key cryptography based on elliptic curves is feasible to be used in sensor networks [16,17,23].

We conducted a comparative performance evaluation of our protocol with the GDH.3 protocol [28] for various network topologies using both experiments and simulation. The experimental study demonstrates the feasibility of implementing our protocol in real sensor network devices while the simulation study highlights the advantages and disadvantages of each approach given the available technology and the corresponding efficiency (energy, time) criteria. Overall our protocol manages to evenly distribute the energy dissipation among the sensor devices, leading to better energy balance.

2 Preliminaries of Elliptic Curve Theory

In this section we review some basic concepts regarding elliptic curves and their definition over finite fields. The interested reader may find additional information in e.g. [6,26]. We also assume familiarity with elementary number theory (see e.g., [11]). The elliptic curves are usually defined over *binary fields* F_{2^m} ($m \geq 1$), or over *prime fields* F_p, $p > 3$. In our experimental results we used elliptic curves defined over prime fields.

An *elliptic curve* $E(F_p)$ over a finite field F_p, where $p > 3$ and prime, is the set of points $(x, y) \in F_p$ (represented by affine coordinates) which satisfy the equation

$$y^2 = x^3 + ax + b \tag{1}$$

and $a, b \in F_p$ are such that $4a^3 + 27b^2 \neq 0$. The set of solutions (x, y) of Eq. (1) together with a point $\mathcal{O}$, called the *point at infinity*, and a special addition operation define an Abelian group, called the *Elliptic Curve group*. The point $\mathcal{O}$ acts as the identity element (for definition of addition see [6,26]).

The *order* m of an elliptic curve is the number of the points in $E(F_p)$. The *order of a point* P is the smallest positive integer n for which $nP = \mathcal{O}$. Application of Langrange's theorem (see e.g. [11]) on $E(F_p)$, gives that the order of a point $P \in E(F_p)$ always divides the order of the elliptic curve group, so $mP = \mathcal{O}$ for any point $P \in E(F_p)$, which implies that the order of a point cannot exceed the order of the elliptic curve.

The security of elliptic curve cryptosystems is based on the difficulty of solving the discrete logarithm problem (DLP) on the EC group. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is about determining the least positive integer k which satisfies the equation $Q = kP$ for two given points Q and P on the EC group. A user A in an elliptic curve cryptosystems can choose a random integer $0 < k < p-1$ and send Q to a user B with whom he wants to communicate secretly. A's public key is Q and his private key is k. Then an encryption algorithm can be applied (e.g. ElGamal encryption [15]) so that B can encrypt the message he wishes

to send to A with the public key Q and A will decrypt it using his private key k.

The Elliptic Curve Diffie-Hellman Algorithm algorithm [13] is based on the difficulty of solving the discrete logarithm problem. Its Elliptic Curve analog is as follows. Let A and B be two entities that wish to share a secret key. Both A and B generate a pair of public/private key (k_A, Q_A) and (k_B, Q_B) respectively. Then A sends Q_A to B and B sends Q_B to A. A computes $S = k_A Q_B$ and B $S = k_B Q_A$, where S is now their shared secret key. The only weakness of this algorithm is that there must be an authentication process between A and B so that there is a guarantee that every entity is who he claims to be. In the group key protocols that will be described in the next section, we assume that there is such an authentication between every two members of the group.

3 The GDH.3 Group Key Establishment Protocol

GDH.3 protocol was presented in [28] together with GDH.1 and GDH.2 protocols. It is particularly suitable for environments which require the minimum computational effort from each group member. The protocol evolves in four stages and here we will present its elliptic curve analog. Suppose that every member in the group agreed on the use of the same elliptic curve parameters. The number of participants is n and we will denote by M_i the i-th participant.

(i) In the first stage every group member M_i generates a random secret value k_i. The M_1 participant selects a point P and sends to M_2 the point $Q_1 = k_1 P$. Then M_2 sends to M_3 the point $Q_2 = k_1 k_2 P$ and so on until the protocol reaches member M_{n-1}. Notice here that the protocol must pass only one time from every participant.

(ii) Group member M_{n-1} computes the point $Q_{n-1} = k_1 k_2 \dots k_{n-1} P$ and sends it to all M_i, with $i \in [1, n]$.

(iii) In the third stage every group member M_i, $i \in [1, n-1]$ computes a point $G_i = k_i^{-1} Q_{n-1}$ and sends it to the last group member M_n.

(iv) M_n calculates the values $k_n G_i$ and send them to the corresponding members M_i.

After these stages, every group member M_i can calculate the group key $Q_n = k_1 k_2 \dots k_n P$ by multiplying the value $k_n G_i$ with its secret number k_i. Despite its efficiency, the disadvantage of GDH.3 protocol is that it does not offer symmetric operation, because all the participants in the protocol do not perform the same number of operations. If the number of the participants is large, then the computational effort in member M_n can be devastating for its energy. For this reason, we propose a new protocol which offers symmetric operation and requires that all devices perform the same number of operations.

4 Our Group Key Establishment Protocol

The GDH.3 protocol [28] relies on communication primitives that provide global ordering of the devices (stage 1), e.g. such as a (virtual) ring-based topology and enable many-to-many message exchanges (stages 2,3). In fixed infrastructure based networks, such communication primitives can be provided by the fixed part (i.e. base stations), however, in wireless sensor networks the fixed infrastructure is sparse (or even non existing), making it difficult (or even impossible) to implement such primitives via external coordination. Certainly one can assume that the participating devices are capable of transmitting at long ranges, allowing them to communicate directly with each other. Still, in the light of the dense deployment of sensor devices close to each other, a traditional single hop communication scheme consumes a lot of power when compared to distributed short-range hop-by-hop propagation ([18]). In addition, multi-hop communication can effectively overcome some of the signal propagation effects in long-distance wireless transmissions and may help to smoothly adjust propagation around obstacles. Finally, the low energy transmission in hop-by-hop propagation may enhance security, protecting from undesired discovery of the data propagation operation.

In the sequel we abstract the technological specifications of existing wireless sensor systems [24] as a system consisting of n devices connected through bidirectional channels allowing direct communication between pairs of neighbor processes linked by a channel. We assume that devices have distinct identities and only local information is available, i.e. they know their own identities together with those of their neighbors but no global information is available, such as the total number n or the structure of the network. To simplify the presentation of the protocol, we here assume that channels are safe, that is, messages are delivered without loss or alteration after a finite delay, but they do not need to follow a first-in-first out rule. In Sec. 5 we implement (in nesC code running in TinyOS) safe communication demonstrating that the underlaying technology can fulfill these assumptions.

We follow a distributed approach that does not require many-to-many message exchanges as in the case of the second and third stages of the GDH.3 protocol and does not rely on any global ordering of the devices. Our protocol is based on the observation that in the first stage of the GDH.3 protocol, group member M_n can compute the shared group key Q_n by acquiring the point Q_{n-1} from M_{n-1} and multiply it with its secret value k_n. Moreover, the points $Q_i = k_1 k_2 \dots k_i P$ which are generated by each group member M_i can be used as their public keys while their private keys are the values k_i. Using this observation, our protocol totally avoids the third and fourth stages of GDH.3 protocol as follows:

(i) In the first stage every group member M_i generates a random secret value k_i. The M_1 participant selects a point P and sends to M_2 the point $Q_1 = k_1 P$ which is its public key and can be used by M_2. Then M_2 sends to M_3 the point $Q_2 = k_1 k_2 P$ (which is the public key of M_2) and so on until the protocol reaches member M_n. The point Q_n is the shared secret key and is calculated by M_n.

(ii) M_n now encrypts Q_n with M_{n-1}'s public key Q_{n-1} and sends it to M_{n-1}. M_{n-1} can decrypt the message with his private key k_{n-1}, acquire the secret value Q_n, encrypt it with the public key of M_{n-2} and send the result to M_{n-2}. The same process will be followed by M_{n-2} and so on, until the protocol reaches member M_1.

In particular, we employ a distributed sequential traversal algorithm that visits each participant of the group in order to build the shared secret key (starting from M_1 and reaching M_n). This is similar to the first stage of GDH.3, without necessarily requiring direct communication among all the group members and still guaranteeing that each participant are visited only one time. When the traversal is finished and all participants are visited, the protocol reaches member M_n that calculates the point Q_n, the shared secret key. Finally, in order for M_n to communicate the shared secret key to all the participants of the group, it repeats the distributed traversal but in reverse order. M_n now encrypts Q_n with M_{n-1}'s public key Q_{n-1} and sends it to M_{n-1}. M_{n-1} can decrypt the message with his private key k_{n-1}, acquire the secret value Q_n, encrypt it with the public key of M_{n-2} and send the result to M_{n-2}. The same process will be followed by M_{n-2} and so on, until the protocol reaches member M_1.

Assigning Groups. Our group key distribution protocol is applicable to hierarchical wireless sensor networks, in the sense that among the nodes there is a hierarchy based on their capabilities. The hierarchical network is composed by a base station, group leaders and simple nodes (group members). The base station represents the authorities of this remote surveillance system (i.e. where the wireless sensors report), has very large storage and data process capabilities and is usually a gateway to another network (i.e. the internet). Typically, the sensors are deployed around the area of the base station and form groups given the needs of the base station. Group leaders are ordinary sensor nodes which can collect local traffic and send it to the base station. Also, they are trusted components and sensors in their groups get routing information from them. Data flow in our network is group-wise within a group of sensor nodes.

Initial Group formation. Based on the initial assignment of the sensors in a given group M (defined by the base station), every group member M_i generates a random secret value k_i while the group leader M_1 selects a point P and calculates the point $Q_1 = k_1 P$ which is its public key. Then, based on a distributed sequential traversal algorithm that visits all partipants at least once, and particularly a distributed *depth-first* traversal algorithm, it sends Q_1 to M_2 (i.e. a neighboring group member) via special `SEARCH<`M_2, Q_1`>` message. When M_2 receives the message, it becomes *active for the first time*, it becomes *visited* and defines participant M_1 as its father (we say M_2 joins the traversal). Moreover, when M_2 is active, it calculates the point $Q_2 = k_1 k_2 P$ (which is the public key of M_2) and shifts the control to a non visited neighbor M_3, through a special `SEARCH<`M_3, Q_2`>` message.

When the protocol reaches member M_u with all its neighbors being visited, M_u encrypts Q_u with M_{u-1}'s public key Q_{u-1} and sends it to M_{u-1} (its father) using

a special PARENT<M_{u-1}, encrypt(Q_u, Q_{u-1})> message [6]. M_{u-1} can decrypt the message with his private key k_{u-1}, acquire the secret value Q_u and either continue the distributed sequential traversal by shifting the control to the next non visited neighbor (if any), again through a special SEARCH<M_v, Q_u> or if all its neighbors have been visited, send a PARENT<M_{u-2}, encrypt(Q_u, Q_{u-2})> message to its parent. This process continues until the protocol reaches the last member of the group M_n that calculates the point Q_n, the shared secret key. In a similar way with M_u, M_n encrypts Q_n with M_{n-1}'s public key Q_{n-1} and sends it to M_{n-1} (its father) using a special PARENT<M_{n-1}, encrypt(Q_n, Q_{n-1})> message.

In the case where M_u has more than one children (in the *depth-first* virtual tree), upon receiving the PARENT message from the last child (let this be M_v) that contains the shared key, it first sends the PARENT<M_{u-1}, encrypt(Q_n, Q_{u-1})> message to its father and then informs its children by sending the special message UPDATE<M_i, encrypt(Q_n, Q_u)>. The child M_i that receives an UPDATE<M_i, encrypt(Q_n, Q_u)> message, decrypts it to acquire the secret key and forwards it to its child by sending an UPDATE<M_{i+1}, encrypt(Q_n, Q_i)> message. This ensures that the shared key will traverse the *depth-first* virtual tree all the way up to the group leader M_1 but also reach all the nodes that belong to a branch of the tree.

Implementation of this traversal technique requires for the active process to know exactly which of its neighbors are visited. To do so, our distributed algorithm apart from the SEARCH and PARENT message, uses a special VISITED message that allows each visited process to inform its neighbors (by broadcasting the message) that it has joined the traversal.

Our protocol essentially builds a depth-first search spanning tree of a network, given a distinguished node as its root (i.e. M_1) and its correctness essentially follows from the correctness of the sequential DFS algorithm, because there is no concurrency in the execution of this algorithm [4].

Handling JoinGroup Events. When a new member M_{n+1} wants to join a group, it must first be authenticated by the base station and get an ID and then contact the group leader (via the nearest group member M_u and through the virtual tree structure) a JOIN<M_u, M_{n+1}> message. The group leader replies by sending the old group key Q_n (again via the tree structure). Then M_{n+1} generates a random value k_{n+1}, computes the new group key $Q_{n+1} = k_{n+1}Q_n$ and sends it back to the group leader. Finally, the group leader sends an UPDATE message to all group members, again by using the virtual tree. The need to contact the group leader is necessary in cases of more than one nodes joining the group simultaneously, in which case, the group leader delays the UPDATE message until all new nodes have joined.

Handling LeaveGroup Events. In the case that a member M_u leaves the group, the group leader generates a random value $\overline{k_n}$ and computes a new group key $\overline{Q_n} = \overline{k_n}Q_n$. Then, as in the case of the *JoinGroup* event, it informs all group members about the new shared by sending an UPDATE message using the virtual tree. However, since the removal of the old member will disrupt the tree structure, the children of

[6] The function encrypt is defined as encrypt(data, key)

M_u must now contact the parent of M_u and update the tree structure. If this is not possible, i.e. because the parent of M_u cannot directly communicate with the children of M_u, the handling of the event fails, and M_1 is signaled to restart the *depth-first* tree construction and generate a totally new shared key.

Handling MergeGroup Events. When a group M' wants to join group M, the procedure followed essentially expands the *depth-first* search tree to include the members of M'. The group leader of M' contacts the leader of M (via the nearest group member of M, M_u) by sending a `MERGE`$<M_u, M'_1>$. The group leader M_1 replies by sending the old group key Q_n (again via the tree structure) to M'_1 that is now denoted as M_{n+1}. When the message reaches M_{n+1}, the distributed sequential traversal algorithm continues as if the members of M' where unvisited members of M. In this sense, M_{n+1} computes a new random value k_{n+1}, calculates the point $Q_{n+1} = k_1 k_2 \dots k_{n+1} P$ (which is now the new public key of M_{n+1}) and shifts the control to the next non visited neighbor M_{n+2} (an old member of M'), through a `SEARCH`$<M_{n+2}, Q_{n+2}>$ message. When all the old members of M' have been visited, the new shared key is propagated to the merged group through the use of the *PARENT* and *UPDATE*.

Handling PartitionGroup Events. Instead of trying to compute a new group key for the two resulting groups, when a *PartitionGroup* event is signaled our protocol simply reconstructs the *depth-first* search spanning tree and generates a new shared key for each group.

Periodic Group Maintenance. We here note that in order to handle the above events, the virtual tree can degenerate into a spanning tree that no longer fulfills the *depth-first* search criteria. Therefore, in order to balance the tree and also in order to guarantee key freshness the group leader periodically restarts the *depth-first* search and generates a new shared key.

Discussion. Our proposed group key protocol satisfies the first two cryptographic properties mentioned in Sec. 1 and in particular the *JoinGroup* event accomplishes forward secrecy while backward secrecy is guaranteed by the *LeaveGroup* event. Regarding the computational group key secrecy this is satisfied since, if an adversary silently overhears radio communication and captures data, he can not discover the group key as it is computationally infeasible to find any secret value k_i from the transmitted data (he has to solve an elliptic curve discrete logarithm problem). Additionally, since our protocol does not require that all group members communicate directly with each other via long-range transmissions. the silent adversary will only be able to listen to a limited number of messages given its actual physical location.

All group events are handled in $O(n)$ time (as in the case of GDH.3, assuming that a bounded number of retransmissions are required due to collisions) and require $O(n)$ message exchanges (again similar to GDH.3, although it is expected that $2n$ less messages need to be transmitted in the network). However, in contrast to GDH.3, our protocol evenly distributes energy consumption among the participants as each device has similar roles in terms of required computations and communication exchanges (energy-wise, the two most demanding events). Balancing the energy dissipation among the sensors in the network avoids the early energy depletion of

	Addition	Multiplication	Random	Encryption	Decryption
Running Time	2.250sec	36.114sec	0.22sec	74.481sec	38.365sec

Table 1
Running times of `EccM-2.0` for operations using 163-bit multiprecision integers

certain sensors (i.e. in GDH.3 participant M_{n-1}) and thus increases the lifetime of the system by preventing from early network disconnection [27]. In contrast to the GDH.3 protocol, our protocol does not assign different roles to the participating devices nor requires some of them to transmit more messages than others. The distributed sequential traversal ensures that the devices consume more or less equal amounts of energy as they perform the same number of events and communication exchanges leading to better energy balance.

5 Performance Evaluation

In order to evaluate the suitability of our protocol in sensor networks we carried out a set of experiments based on the MICA2 mote architecture [24]. Currently, these devices represent the state of the art in wireless sensor networks technology based on commercial off-the-shelf hardware components and offer an 8-bit, 7.3 MHz ATmega 128L processor, 4 KB of primary memory (RAM) and 128 KB of program space (ROM) and 512 KB secondary memory (EEPROM) and a ChipCon CC1000 radio capable of transmitting at 38.4 KBps powered by 2 AA batteries. In software, we implemented our protocol using the nesC programming language and work with the Elliptic Curve Cryptography module EccM [23], implemented specifically for TinyOS, that also allows to represent and carry out basic operations with multiprecision integers of 160-bit size. Given this particular selection of hardware/software we evaluated the running times for generating random secret values k_i, multiplying the secret values with a point P and encrypting/decrypting them based on a given set of public/private keys. The running times shown in Table 1 were measured by the MICA2 device using the `SysTime`, TinyOS component that provides a 32-bit system time based on the available hardware clock. To get good average results, we allowed the device to repeat each operation at least 100 times. The experimental results indicate that elliptic curves implementation is feasible in sensor devices, as time to perform an encryption and decryption averages out to 74.481sec and 38.365sec.

Given the above running times for performing the necessary cryptography operations, we continue by conducting a comparative evaluation study on the performance of GDH.3 and our protocol via simulation. The experimental evaluation is conducted with Power-TOSSIM [25] that simulates the wireless network at the bit level, using TinyOS component implementations almost identical to the MICA2 CC1000-based radio stack. In this set of experiments, the amount of time spent executing instructions is not captured by TOSSIM. We generated loss rates from various different physical topologies of wireless sensor networks using the `LossyBuilder` tool provided by TOSSIM [21]. The transmission range of the devices was set to 50

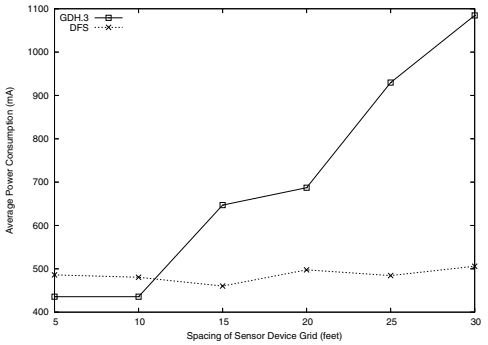

Fig. 1. Average energy dissipation per sensor device for each protocol, in a 7 by 7 grid

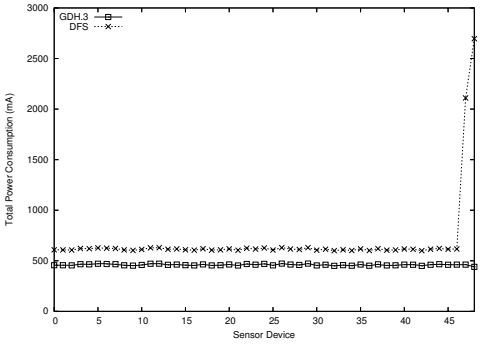

Fig. 2. Energy dissipation per sensor device for each protocol, in a 7 by 7 grid with node spacing set to 15

feet and the topologies considered are 5 by 5 ($n = 25$), 7 by 7 ($n = 49$) and 9 by 9 ($n = 81$) grids with spacing of $\{5, 10, 15, 20, 25, 30\}$ feet ($45'$ by $45'$).

Our protocol (see Sec. 4) relies on certain assumptions on the underlaying network. In order to provide the necessary high-level primitives for the protocol to be operational we implement two additional modules: (i) the `Reckon` module that provides information on the identities of the neighboring nodes (that belong in the same group with the node) based on short `HELLO` messages and (ii) the `SafeSend` module that guarantees that messages are finally delivered to their destinations by periodically retransmitting messages until the destination confirms their safe reception. The GDH.3 (see Sec. 3) requires all devices to communicate directly with M_{n-1}. To be able to provide this primitive, we implemented the `Flood` module that simply floods the messages in the network until they are received by their final destination. We here note that the `Bcast` module provided by TinyOS implements many-to-1 multi-hop routing and is thus unsuitable. Finally, since in GDH.3 the devices need to know identities of all the group members (so that stage 1 can be carried out) we also implemented the `ReckonGlobal` module that provides information on the identities of all the nodes (that belong in the same group with the node) based on a simple flooding protocol of `HELLO` messages.

The experimental results shown in Fig. 1 indicate the differences of the two protocols. When the spacing of the devices is small and all devices can communicate directly with each other, the GDH.3 protocol consumes less energy than our protocol. However, as the spacing of the nodes increases and the devices can no longer communicate directly with each other, GDH.3 starts to spend more energy. On the other hand, our protocol energy consumption is not affected by the spacing of the devices since each device needs to communicate with its neighbors. We here note that similar results hold for other grid sizes (see Fig. 3, 4). Furthermore, based on Fig. 3, 4 it is clear that both protocol spend more energy as the grid size (and hence n) increases, especially when the spacing between the devices is big. Regarding the energy balance achieved by the two protocols, Fig. 2 clearly depicts the exhessive power spent by nodes M_{n-1} and M_n that both have to transmit $n-2$ and $n-1$ messages during the second and fourth phases of the GDH.3 protocol.

Based on the above observations, it seems that GDH.3 is more suitable in cases

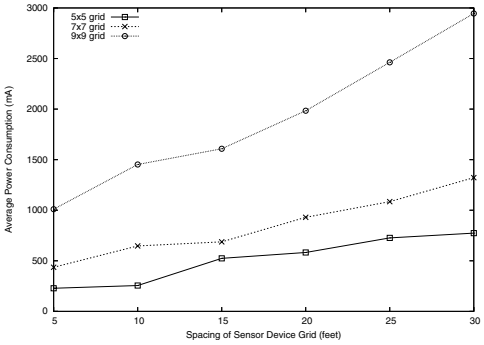

Fig. 3. Average energy dissipation per sensor device when executing the GDH.3 protocol, for different node spacing and grid size

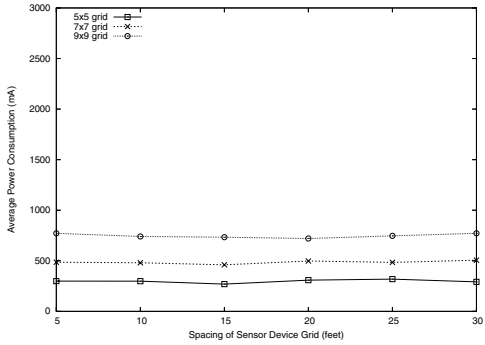

Fig. 4. Average energy dissipation per sensor device when executing our protocol, for different node spacing and grid size

when the network size is small and all devices are capable of communicating directly with each other. On the other hand, for sparse networks of large sizes it seems that our protocol is more suitable. Furthermore, since our protocol requires a large number of encryption/decryption operations to be carried out during the distributed sequential traversal, the overall time to compute the shared secret key is larger than the corresponding time required by GDH.3. On the other hand, our protocol distributes the energy dissipations to all participating nodes evently leading to better energy balance.

6 Conclusions & Future Work

We have presented a new distributed group key establishment protocol suitable for energy-constrained sensor networks of limited communication capabilities that reduces the overall number of message exchanges by imposing a limited number of additional computations that are however lightweight and easy to implement in current sensor technology. We plan to further investigate the performance of our protocol in case of more frequent JoinGroup/LeaveGroup events and also provide mechanisms that update the underlying tree structure by periodically balancing the tree instead of rebuilding it.

References

[1] Akyildiz, I., W. Su, Y. Sankarasubramaniam and E. Cayirci, *Wireless sensor networks: a survey*, Journal of Computer Networks **38** (2002), pp. 393–422.

[2] Amir, Y., Y. Kim, C. Nita-Rotaru and G. Tsudik, *On the performance of group key agreement protocols*, ACM Transactions on Information and Systems Security **7** (2004), pp. 457–488.

[3] Anton, E. and O. Duarte, *Performance analysis of group key establishment protocols in ad hoc networks*, Technical report, Universidade Federal do Rio de Janeiro, Brazil (2006), technical Report GTA-03-06.

[4] Attiya, H. and J. Welch, "Distributed Computing: Fundamentals, Simulations and Advanced Topics," McGraw-Hill Publishing Company, 1998.

[5] Becker, K. and U. Wille, *Communication complexity of group key distributionn*, in: *5th ACM Conference on Computer and Communications Security (CCS 1998)* (1998), pp. 1–6.

[6] Blake, I., G. Seroussi and N. Smart, *Elliptic curves in cryptography*, Technical report, Cambridge University Press (1999), london Mathematical Society Lecture Note Series 265.

[7] Bresson, E., O. Chevassut and D. Pointcheval, *Provably authenticated group diffie-hellman key exchange - the dynamic case*, in: *Advances in Cryptology (ASIACRYPT 2001)* (2001), pp. 290–309, lecture Notes in Computer Science, LNCS 2248.

[8] Bresson, E., O. Chevassut and D. Pointcheval, *Dynamic group diffie-hellman key exchange under standard assumptions*, in: *Advances in Cryptology (EUROCRYPT 2002)* (2002), pp. 321–336, lecture Notes in Computer Science, LNCS 2332.

[9] Bresson, E., O. Chevassut, D. Pointcheval and J. Quisquater, *Provably authenticated group diffie-hellman key exchange*, in: *8th ACM Conference on Computer and Communications Security (CCS 2001)* (2001), pp. 255–264.

[10] Burmester, M. and Y. Desmedt, *A secure and efficient conference key distribution system*, in: *Advances in Cryptology (EUROCRYPT 1994)* (1994), pp. 275–286, lecture Notes in Computer Science, LNCS 950.

[11] Burton, D., "Elementary Number Theory," McGraw-Hill Publishing Company, 1998, 4th edition.

[12] Chatzigiannakis, I., A. Kinalis and S. Nikoletseas, *Wireless sensor networks protocols for efficient collision avoidance in multi-path data propagation*, in: *ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN 2004)*, 2004, pp. 8–16.

[13] Diffie, W. and M. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **22** (1976), pp. 644–654.

[14] ElGamal, T., *A conference key distribution system*, IEEE Transactions on Information Theory **28** (1982), pp. 714–720.

[15] ElGamal, T., *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory **31** (1985), pp. 469–472.

[16] Gaubatz, G., J. Kaps and B. Sunar, *Public key cryptography in sensor networks – revisited*, in: *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)* (2004), pp. 2–18, lecture Notes in Computer Science, LNCS 3313.

[17] Gura, N., A. Patel, A. Wander, H. Eberle and S. Shantz, *Comparing elliptic curve cryptography and RSA on 8-bit CPUs*, in: *Cryptographic Hardware and Embedded Systems (CHES 2004)* (2004), pp. 119–132, lecture Notes in Computer Science, LNCS 3156.

[18] Heinzelman, W. R., A. Chandrakasan and H. Balakrishnan, *Energy-efficient communication protocol for wireless microsensor networks*, in: *33rd IEEE Hawaii International Conference on System Sciences (HICSS 2000)*, 2000, p. 8020.

[19] Katz, J. and M. Yung, *Scalable protocols for authenticated group key exchange*, in: *Advances in Cryptology (CRYPTO 1994)* (1993), pp. 110–125, lecture Notes in Computer Science, LNCS 2729.

[20] Kim, Y., A. Perrig and G. Tsudik, *Tree-based group key agreement*, ACM Transactions on Information and Systems Security **7** (2004), pp. 60–96.

[21] Levis, P., N. Lee, M. Welsh and D. Culler, *TOSSIM: Accurate and scalable simulation of entire tinyos applications*, in: *1st ACM International Conference On Embedded Networked Sensor Systems (SENSYS 2003)*, 2003, pp. 126–137.

[22] Liao, L., "Group Key Agreement for Ad Hoc Networks," Master's thesis, Ruhr-University Bochum, Germany (2005).

[23] Malan, D., M. Welsh and M. Smith, *A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography*, in: *2nd IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, 2004, pp. 71–80.

[24] *Crossbow Technology Inc. MICA2 motes* (2005), http://www.xbow.com/Products/productsdetails.aspx?sid=72.

[25] Shnayder, V., M. Hempstead, B. Chen, G. Allen and M. Welsh, *Simulating the power consumption of large-scale sensor network applications*, in: *2nd ACM International Conference on Embedded Networked Sensor Systems (SENSYS 2004)*, 2004, pp. 188–200.

[26] Silverman, J., "The Arithmetic of Elliptic Curves," Springer Verlag, 1986.

[27] Singh, M. and V. Prasanna, *Energy-optimal and energy-balanced sorting in a single-hop wireless sensor network*, in: *1st IEEE International Conference on Pervasive Computing and Comminications (PERCOM 2003)*, 2003, pp. 50–59.

[28] Steiner, M., G. Tsudik and M. Waidner, *Diffie-Hellman key distribution extended to group communication*, in: *3rd ACM Conference on Computer and Communications Security (CCS 1996)* (1996), pp. 31–37.