# Batch-based Group Key Management with Shared Key Derivation in the Internet of Things

Luca Veltri, Simone Cirani, and Gianluigi Ferrari
Department of Information Engineering
University of Parma, Italy
Email: {luca.veltri, simone.cirani, gianluigi.ferrari}@unipr.it

Stefano Busanelli
Guglielmo Srl
Pilastro di Langhirano, Parma, Italy
Email: stefano.busanelli@guglielmo.biz

***Abstract*—Many applications for ad-hoc networks are based on a multicast communication paradigm, where a single source sends common data to many receivers. In these contexts, it is possible to efficiently secure the multicast communications by leveraging on a common secret key, denoted as "group key", shared by multiple users. In this paper, we propose a novel centralized approach that efficiently addresses the problem of deriving and managing a group key in generic ad-hoc networks and Internet of Things (IoT) scenarios, reducing the computation overhead due to group membership changes caused by user's joins and leaves. In particular, the proposed method takes advantage of the assumption of two possible leave strategies: (i) at a pre-determined time interval selected when the member joins the group or (ii) at any unpredictable time interval, as in the case of membership revocation.**

***Keywords*—Key distribution, Multicast communication, Security, Internet of Things***

## I. Introduction

According to a group communication paradigm, a single member can originate and deliver a message to the whole group of nodes, through multicast (or broadcast) communication services [1], and thus in a more efficient manner than an equivalent unicast-based solution. The first applications taking benefit of the group communications model, such as online gaming and audio/video streaming [2], have historically operated on the Internet. In recent years, the ever increasing diffusion of ad-hoc (mostly wireless) networks has offered a new fertile ground for the development of new types of group-based applications. In scenarios such as wireless sensor networks [3], mobile ad-hoc networks, and Internet of Things (IoT) [4], a large number of applications (e.g., data dissemination, data gathering, peer-to-peer communications) need an underlying multicast data delivery service.

Securing group communications consists in providing confidentiality, authenticity, and integrity of messages exchanged within the group, through suitable cryptography services [5], and without interfering with the data path of the multicast data flow[1] [7]. The achievement of this goal in an efficient and scalable manner is a challenging task since it requires that a large and dynamically varying number of users share cryptographic materials, even in the presence of unpredictable group membership changes due to new users entering (joining) the network and to old users leaving the network. In fact, after any membership change, the shared cryptographic materials should be refreshed through a suitable *rekeying* operation, so that a former group member has no access to current communications (*forward secrecy*) and a new member has no access to previous communications (*backward secrecy*) [8], [9].

While authenticity and integrity protection in group communications can be easily achieved through asymmetric cryptography, like in traditional point-to-point communications (e.g., through digital signature), the simplest and most scalable way to provide data confidentiality within a multicast group is to encrypt the data through symmetric cryptography, with a secret key shared (only) by all users belonging to the group. Such symmetric key is normally referred to as *group key*.

Distributing such secret group key to all the legitimate users and updating it at any group membership change is a problem known as *Group Key Distribution* or *Multicast Key Distribution* (MKD) [10]. There are two main categories of MKD protocols: centralized [11] or distributed [12]. According to the former, the keys' distribution task is assigned to a single entity, denoted as Key Distribution Center (KDC). In the case of the distributed approach instead, the group key is established and maintained by the users themselves, in a distributed fashion. The centralized MKD approach has several advantages: (i) simplicity; (ii) a small number of exchanged messages compared to other methods; (iii) the possibility of operating on intrinsically broadcast channels, where the source (which also acts as MKD server) sends data to all the possible destinations. Distributed methods typically offer greater reliability, since they do not require any centralized entity to trust, but they have higher communication and computational costs and are not applicable to asymmetric communication scenarios where data cannot be exchanged between any pair of nodes. For these reasons, in the rest of this paper we will focus on centralized approaches.

The technique described in this work is based on a key derivation scheme properly extended in order to deal with both unpredictable leave events and collusive attacks. In particular, we present a MKD protocol tailored for very dynamic ad-hoc networks, either wired or wireless. Time is partitioned in fixed-length intervals, each of them associated with a different group key. Even if a user can join anytime (asynchronously), it shall wait until the beginning of the next slot before becoming a group member. This introduces a delay, on average

[1]Iolus [6], for example, is a scheme that interferes with the normal packet stream, since a group security intermediary has to decrypt and encrypt all the packets transiting in its own group.

equal to half of the slot interval, but allows to reduce the number of rekeying acts. Similarly, the planned leave of a legitimate member shall also happen at the beginning of a slot period. In other words, the protocol is slotted and adopts a synchronous *batch rekeying* mechanism [13] that improves efficiency without posing security threats. The protocol also provides proper mechanisms to deal with unpredictable leave events and also to resist against collusive attacks but only in case no eviction occurs.

The aim of the protocol is to minimize the computational burden of group members and the overhead, expressed in terms of number of exchanged messages, while achieving a sufficiently high security level. The proposed protocol can operate on very dynamic scenarios with a large number (thousands) of nodes and offers excellent performance under the assumption of low rate of evictions.

The structure of this paper is as follows. Section II reviews related work. In Section III, a new technique for solving the problem of distributing group-shared secret keys for ad-hoc communications is proposed and its performance evaluated in relation to state-of-the-art protocols. In Section IV, the new technique is applied to a realistic IoT application scenario. Finally, concluding remarks are given in Section V.

## II. Related Work

In multicast group communications, a proper MKD protocol is required for generating and distributing a secret group key the can be used to secure (encrypt) data sent from one source to all destinations that are member of the same group. Since multicast groups are often very dynamic, due to the join of new members and the leave of old members, the MKD has to handle such group membership changes by re-generating and re-distributing new group keys.

As described in Section I, in a common centralized MKD scenario, the KDC may share an individual long-term secret key with every user of the network, while a shared short-term key is used as group key and refreshed after any membership change (or programmatically) using the long-term keys. However, this plain centralized solution is not scalable since the number of communications required for the rekeying operation is *linear* in the size of the current group (denoted as $n$). These results should be compared with the known lower bound $O(\log_2 n)$ [9].

Wong et al. [8] proposed the Logical Key Hierarchy (LKH) approach, based on key graphs, where keys are arranged into a hierarchy, and the key server maintains all the keys. The LKH scheme makes use of symmetric-key encryption (as the only cryptographic primitive), and has a number of communications approaching the lower bound in [9].

If a user wants to join the group, it sends a join request to the key server. The user and key server mutually authenticate each other using a protocol such as Secure Socket Layer (SSL). If authenticated and accepted into the group, the user shares with the key server a symmetric key, called the user's individual key.

In [14], the authors propose the MARKS protocol, which is scalable and requires no key update messages. However, MARKS only works if the leaving time of a member is set when the member joins the group, so that members cannot be expelled. Besides the scheduled leaves, there is also the possibility of unpredictable leaves, which occurs when a user is evicted from the group. In this case, it is unsafe to delay the rekeying until the next time slot, and it is necessary to provide a mechanism which allows immediate revocation of all the cryptographic materials known by the evicted user.

In a previous work [15], we presented a very simple algorithm for key derivation, which however does not specify any management scheme to deal with unpredictable leave events and does not protect against collusive attacks. In this work, in order to allow rapid unpredictable evictions, we superimpose an existing asynchronous key management mechanism, denoted as Logical Key Hierarchy (LKH) [8], to our slotted protocol.

## III. New Group Key Management Mechanism

In this section, a new group key distribution mechanism is presented. The proposed mechanism allows a server (KDC) to efficiently distribute a group key to all members of a multicast group dealing with dynamic joins and leaves of users as group members. The proposed solution is first summarized in Subsection III-A and then detailed in Subsections III-B and III-C.

### A. Protocol Overview

Let us consider a multicast group communication scenario in which the same data has to be securely sent to a group of destinations. In order to guarantee data confidentiality, the sent message has to be encrypted with a secret (group) key shared by, and only by, all group members. We consider a dynamic scenario in which, at any time, a new user may join the system as new group member and an old user may leave the group. As described in the previous sections, this requires a suitable group key distribution mechanism, able to distribute a new key to all members upon every change of group membership. We consider a key distribution scenario based on a trusted KDC that takes care of: (i) maintaining a secure association with all users belonging to the system; (ii) generating a new group key every time the group membership changes; (iii) efficiently managing the distribution of the new group key to all group members, guaranteeing both forward and backward secrecy.

In a more general scenario, join and leave operations occur unpredictably, in a completely asynchronous and dynamic way. However, in order to optimize and significantly reduce the complexity and the number of exchanged messages required to handle group member changes and group key re-distribution (rekeying), a more practical method is to allow the KDC to handle simultaneously a number of membership changes. This can be achieved by splitting time into intervals (sometimes referred to as "time slots" or, simply, "slots") and letting the KDC handle all membership changes that occur in the same time interval. Key distribution mechanisms that work in this way are often referred to as "batch" methods. Note that our proposed method applies when these time intervals have the same length or different lengths. However, very common scenarios are those in which membership changes are handled, for practical reasons, in a daily or monthly manner: this is the case, for example, of applications that consider service subscriptions with specific durations (expressed exactly in days or months). Other common possible time slot units can be minutes, seconds, or years.

Although the time slot in which a new user wants to join the system is in general difficult (or impossible) to predict (as it can apply at any time), there are many application scenarios in which the duration of the membership of a user is specified at the moment when the user joins the system, possibly further extended on the basis of a renewal strategy. Service subscriptions are often handled by applications in this way, with the possibility (in a limited subset of cases) of considering some form of revocation mechanisms in order to handle situations (often seen as exceptions) in which a membership has to be revoked in advance before its natural expiration time (for example, if a user unexpectedly leaves the system or if he/she is removed due to a misuse or for administrative reasons).

In spite of the above considerations, the majority of the proposed key management mechanisms do not take advantage of this operation and simply consider any leave event as not pre-determined, as it always occurs randomly.

On the opposite, we explicitly consider two different kinds of leave events: (i) "pre-determined" leave events, when the leave time is selected in advance when the user joins the network or when it refreshes his/her membership, as in the case of a natural membership expiration; (ii) "unpredictable" leave events, when the time of leave does not coincide with the one selected at the time of joining or refreshing, for example in the case of explicit membership revocation. In our method, like in [14], both kinds of leave events are explicitly considered, taking the advantage of the balance of the former leaving strategy with respect to the latter.

We consider a different group key $K_i$ for each time slot $i$ with $i = 0, 1, 2, \ldots, N$. In order to efficiently handle both kinds of leave events, the group key is obtained through a one-way function of two sub-keys $K1_i$ and $K2$:

$$K_i \doteq f(K1_i, K2) \qquad i = 0, 1, 2, \ldots$$

with $K1_i$ and $K2$ properly managed in order to handle both kind of leaves. In particular, the values $K1_i$ are associated to every time slots $\Delta t_i$ (with $i = 0, 1, 2, \ldots, N$); they are pre-determined and provided to group members according to their assigned membership duration. The values of $K1_i$ are generated in an intelligent and secure manner in order to simplify the assignment to joining users, by providing only some root secret materials that can be used by the member to further derive all $K1_i$ values associated with all time slots he/she subscribed for. $K1_i$ are then used to handle all new join and "pre-determined" leave events.

On the other hand, $K2$ is used to handle all "unpredictable" leave events. It is changed and re-distributed by the KDC to all (and only) group members, in a scalable way, similarly to other mechanisms already proposed in the literature.

Since the amount of operations and exchanged messages differ for managing of the sub-keys $K1_i$ and $K2$, the total amount of operations and exchanged messages is a function of the rate of the "unpredictable" leave events over join and "pre-determined" leave events.

Details of how $K1_i$ and $K2$ are derived and managed are hereafter described.

## B. Protocol Details

The objective of the proposed key management protocol is to provide a group key that can be securely shared by (and only by) all group members, taking into account and properly handling:

1) regular membership changes, that are due to new users that join the group and active members that leave the group for "clean" membership expiration ("pre-determined" leaves);
2) exceptional active member leaves, e.g., in the case of explicit membership revocation ("unpredictable" leaves).

In order to take into account membership changes of type 1, the overall time span is considered divided into a sequence of $N$ time slots $\Delta t_i$ with $i = 0, 1, 2, \ldots, N$ and in general, $\Delta t_i \neq \Delta t_j$ for $i \neq j$. In practice, however, it will be common to have $\Delta t_i = \Delta t_j = \Delta t \; \forall i, j$, with $\Delta t$ equal to standard time units, such as a minute, a second, a month, etc. For each time interval $\Delta t_i$, in the following referred as "slot" $i$, a different group key $K_i$ is determined. Consider now a user member $x$ that will belong to the group from time $t_a$ to time $t_b + 1$, i.e. from time slot $\Delta t_a$ to time slot $\Delta t_b$: he/she will receive the subset of keys $S_X = \{K_i, \text{ with } i = a, a+1, a+2, \ldots, b\}$.

According to the above approach, as far as only membership changes of type 1 are considered, the KDC is requested to generate all keys $K_i$ and give to each new incoming member only the subset of keys corresponding to the time slots over which he/she will belong to the group. If the member will stay for a total of $m$ time slots, this will require the KDC to give to the new member $m$ different keys. In order to limit the total amount of cryptographic material that the KDC has to send to each new member, a proper distribution protocol is adopted.

However, regular membership changes (type 1) are not the only events that require the assignment and distribution of a new group key (i.e., a rekeying operation). In the case of an unpredictable leave event (type 2) in time slot $\Delta t_h$ of member $y$ that negotiated with KDC a membership from time slot $\Delta t_a$ to time slot $\Delta t_b$, at least all previously assigned keys (from $K_h + 1$ to $K_b$) must be re-assigned and distributed to all valid group members. This is needed in order to prevent $y$ to decrypt messages that are sent after time slot $\Delta t_h$ with valid keys that he/she received by the KDC in joining the group.

To handle both types of membership changes in a secure and flexible way, the following key derivation and distribution mechanism is proposed.

Let us consider $N$ time slots, with $N = 2^D$. Each time slot $\Delta t_i$ is associated with a key $K_i$ defined as:

$$K_i \doteq f(K1_i, K2) \qquad i = 0, 1, 2, \ldots, N-1$$

where $K_i$, $K1_i$, and $K2$ are fixed or variable-length bit strings, and $f(,)$ is a cryptographic one-way function that returns a bit string of length equal to or greater than $K_i$. If $f(.)$ returns a bit string of length greater than $K_i$, a truncation can be applied. A cryptographic hash function $H()$ (for example SHA1 or MD5) can be used in place of $f(.)$ as follows:

$$K_i \doteq f(K1_i, K2) = H(K1_i \| K2) \quad i = 0, 1, 2, \ldots, N-1$$

The sub-key $K1_i$ is defined as follows. Consider a binary tree with depth equal to $D+1$, including the root node (level 0). At any level $h$, starting form 0, the binary tree has $2^h$ nodes. The last level is $D$, leading to $2^D = N$ leaves. Let's indicate with $(h, j)$ the node $j$ of level $h$, with $0 \leq h \leq D$ and $0 \leq j \leq 2^h - 1$. Each node $(h, j)$ of the tree, excluding the last level $D$, has two child nodes that are respectively: left child $(h+1, 2j)$ and right child $(h+1, 2j+1)$. Each node $(h, j)$ is associated to a value $x_{h,j}$ that is derived by the value of parent node as follows:

$$\begin{aligned} x_{h+1,2i} &\doteq f_0(x_h, i) \\ x_{h+1,2i+1} &\doteq f_1(x_h, i) \end{aligned}$$

or equivalently:

$$x_{h,i} \doteq \begin{cases} f_0(x_{h-1,i/2}) & i = 0, 2, 4, \ldots, 2^h - 2 \\ f_1(x_{h-1,(i-1)/2}) & i = 1, 3, 5, \ldots, 2^h - 1 \end{cases}$$

where $f_0()$ and $f_1()$ are two different cryptographic one-way functions. They could be also defined based on the same function $f()$ as follows:

$$\begin{aligned} f_0(x) &\doteq f(x) \\ f_1(x) &\doteq f(x+1) \end{aligned}$$

In this case we can write $x_{h,i}$ (recursively) as:

$$x_{h,i} \doteq f(x_{h-1,\lfloor i/2 \rfloor} + (i \bmod 2))$$

By repeatedly applying the previous equations, starting from the value $x_{h,i}$ of node $(h, i)$ it is possible to generate all values associated to the nodes of the sub-graph that has $(h, i)$ as root. At the same time the value $x_{h,i}$ of node $(h, i)$ can be obtained from the value associated to any node along the path from the $x_{h,i}$ to the tree's root $(0, 0)$.

Given such a binary tree, we define the sub-key $K1_i$ equal to the value of the leaf $i$, that is:

$$K1_i \doteq x_{D,i}$$

Then, $K1_i$ can be obtained from the value associated to any node along the path from the leaf $i$ to the tree's root, or equivalently from any values from $x_{D,i}$ to $x_{0,0}$. At the same time, starting the value $x_{h,i}$ of node $(h, i)$ it is possible to obtain all sub-key values in the interval from $2^{D-h} \cdot i$ to $2^{D-h} \cdot (i+1) - 1$ included, that is all sub-keys from $K1_{2^{D-h} \cdot i}$ to $K1_{2^{D-h} \cdot (i+1)-1}$. Note that, as a special case, the value $x_{0,0}$ can generate all sub-keys from $K1_0$ to $K1_{N-1}$.

This property, can be used by the KDC to distribute the $K1_i$ sub-keys to new members in a very efficient way, reducing from $O(N)$ to $O(log(N))$ the number of values that the KDC has to pass to a new member in order to set sub-keys for all the temporal period that the new member will belong to the group.

The worst case occurs when the node joins the group from time slot 1 to time slot N-1, included. In this case, $2 \cdot (log_2(N) - 1)$ keys need to be distributed: $x_{D,1}$, $x_{D,N-1}$, $x_{D-1,1}$, $x_{D-1,\frac{N}{2}-1}$, $\ldots x_{2,1}$, $x_{2,2}$.

Let's now consider the sub-key $K2$. The value $K2$ is maintained constant as far as only regular membership changes happen. As soon as a unpredictable leave event occurs, all un-expired keys of the leaving member must be revoked and replaced by new ones. This objective is reached by replacing the $K2$ that in turn will change all successive group keys $K_i$ that are generated by the values of $K1_i$ and $K2$.

When a new $K2$ value is generated, this has to be distributed by the KDC to all remaining valid group members. This operation is very similar to the one faced by current centralized key distribution protocols: for example (LKH) [8] can be used.

### C. Managing Keys for Unlimited Time Intervals

The described protocol assumes that the number of time slots is fixed and equal to $n = 2^D$. Therefore, it is inevitable that the key distribution mechanism is doomed to come to an end eventually. In this section we sketch a simple extension of the protocol to allow the KDC to handle unlimited time intervals, without requiring each key distribution tree to be kept in memory but just a slightly increased computational effort.

Time is split into intervals $I_k$ of length $\Delta T$ and periods $\Pi_i$ of length $n \cdot \Delta T$. Each period $\Pi_i$ is associated with a given seed $s_i$, which is equal to the value of the root $x^i_{0,0}$ of a tree. Within a given period, the protocol works exactly as described above. If a subscription lasts beyond the end of a period, it is necessary to distribute keys from more then one tree. Each tree can be computed "on the fly" in the following way:

$$x^i_{0,0} \doteq g(s_i) \doteq g(h(s_{i-1}))$$

where $s$ is a seed, $h(.)$ is a *"one-way"* function (i.e. hashing function), and $g(.)$ is a *"blinding"* function (i.e. XOR function). For instance, possible choices for $h(.)$ and $g(.)$ are

$$\begin{aligned} h(s_i) &= H(s_{i-1}) = H^{i+1}(s) \\ g(s_i) &= s \oplus h(s_i) = s \oplus H^{i+1}(iv) \end{aligned}$$

where $H$ is a hashing function and $iv$ is an initial vector.

The key $x_{h,i}$ can be calculated as

$$x_{h,i} = x^P_{h,I} = \underbrace{f(f(\ldots f(}_{h \text{ times}} g(s_{P-1}) + \underbrace{a_0) + a_1) \ldots) + a_{h-1})}_{h \text{ bits}}$$

where $P = \lfloor i/2^h \rfloor$ is the index of the period, $I = i \bmod 2^h$ is the index of the period's interval, and $a_0, a_1, \ldots, a_{h-1}$ are the $h$ bits of the binary representation of $I$.

### D. Performance Evaluation

The performance of the proposed key distribution mechanism is compared with current state-of-the-art solutions. The performance of the mechanism can be evaluated in terms of the following metrics:

- amount of cryptographic material to be sent to a particular node or group members (K1 and K2 sub-keys);
- number of messages to be sent within the group, either from the KDC or relayed by group members.

When a node joins the group, it must receive both the current K2 sub-key and the set of information needed to generate all the K1 sub-keys for its membership interval, which has an upper bound of $2 \cdot (\log_2(N) - 1)$ over a single period. This information can be sent in a single message and is the same amount of information that is distributed in MARKS.

When a node leaves the group gracefully, no information needs to be transmitted. In such a case, the new protocol still performs as MARKS, which requires no communication, while it provides better results than LKH, which requires a re-organization of the nodes' topology.

Finally, in the case of node eviction, the new protocol requires only the K2 sub-key to be retransmitted. The performance in such a case depends exclusively on the distribution strategy adopted. For instance, given a group of $n$ members, the transmission of unicast messages requires $n - 1$ messages. In order to achieve better results, we superimpose the LKH distribution strategy, thus providing the same performance as LKH. As already said in Section II, MARKS only works if the leaving time of a member is set when the member joins the group and therefore it does not apply to the case of unpredictable leave events, such as evictions.

The proposed key distribution protocol offers the best performance of MARKS and LKH in case of join and leave events, respectively. Moreover, our approach takes into account the case of node eviction, thus offering a more exhaustive coverage of the events that might occur in the group's life cycle. Better performance derives from the fact that only in case of key revocation events explicit communication between KDC and group member is required.

## IV. An Application for Data Aggregation in IoT

As aforementioned in Section I, the scheme proposed in this manuscript can be employed in several types of ad-hoc networks. In this section, we will present a case study for data aggregation in the IP-based Internet of Things (IoT).

In-network data aggregation in wireless sensor networks consists in executing certain operations (such as sum and average) on intermediate nodes in order to minimize the amount of transmitted messages and processing required on nodes so that only significant information is passed along in the network. This leads to several benefits, i.e. energy saving, which are crucial for constrained environments, such as low-power and lossy networks. Data aggregation is a multipoint-to-point communication scenario, which requires intermediate nodes to operate on received data and forward a function of such input data. In those scenarios where privacy on transmitted data is an issue, it might be required to send encrypted data. In order to do so, a group key can be used to encrypt sent data and decrypt received data prior to aggregation in a much more efficient scheme than employing a different key between each couple of nodes.

In this work, we applied the proposed scheme to a particular application scenario where wireless sensors running the Contiki OS [16]. Contiki is an open source operating system for the Internet of Things. Contiki allows tiny, battery-operated low-power systems communicate with the Internet. Contiki is used in a wide variety of systems such as city sound monitoring, street lights, networked electrical power meters, industrial monitoring, radiation monitoring, construction site monitoring, alarm systems, and remote house monitoring.

Contiki includes the uIP TCP/IP stack that provides Contiki with TCP/IP networking support. uIP provides the protocols TCP, UDP, IP, and ARP. Secure communication is integrated in the Contiki with a lightweight implementation of the IPSec protocol [17]. IPSec in Contiki requires a hardcoded encryption key to be used, which is therefore static. In this work, a dynamic encryption key configuration mechanism has been implemented in order to make it possible to integrate the Contiki IPSec with our key distribution protocol.

In this scenario, sensor nodes send sensed data to a collecting node securely using IPSec. The presence of a KDC which can communicate securely with each participating node is assumed. The proposed protocol makes it possible to insert and remove nodes seamlessly. This ensures that only authorized nodes are able to exchange data invisible to eavesdroppers. Eavesdroppers are also prevented by trying brute force attacks, as the encryption key is changed automatically over time even if no change in the group membership occurs.

## V. Conclusions

In this paper, we have presented an innovative mechanism for distributing a secret group key to all group members, in a dynamic scenario in which members join and leave the multicast group. The proposed protocol, differently from the majority of the current mechanisms, considers both the cases in which a member leaves the group in a predictable manner, for example for membership subscription expiration, or in a unpredictable manner, such as in case of membership revocation, while ensuring both backward and forward secrecy. In order to optimize and significantly reduce the number of exchanged messages required for handling group member changes and group key re-distribution (rekeying), time is split into time-intervals, thus letting the KDC to handle together all membership changes that occur in the same time interval (batch method). For each time interval a new key is automatically derived by all active members, without any interaction with the KDC. In such way both join and predetermined leave event can be easily handled. Only in case of key revocation events explicit communication between KDC and group member is required, allowing the new protocol to better perform than current key distribution protocols. The proposed scheme has been integrated in an application scenario where Contiki OS-based sensor nodes disseminate sensed data securely using IPSec. This approach makes it is possible to provide group-level confidentiality and integrity, together with per-node authentication and non-repudiation.

## Acknowledgment

The work of Luca Veltri, Simone Cirani, and Gianluigi Ferrari is funded by the European Community's Seventh Framework Programme, area "Internetconnected Objects", under Grant no. 288879, CALIPSO project - Connect All IP-based Smart Objects. The work reflects only the authors views; the European Community is not liable for any use that may be made of the information contained herein.

## References

[1] S. Deering, "Host extensions for IP multicasting," no. RFC 1112, August 1989, the Internet Engineering Task Force (IETF).

[2] T. Turletti and C. Huitema, "Videoconferencing on the Internet," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 340–351, June 1996.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[4] E. Schoch, F. Kargl, M. Weber, and T. Leinmuller, "Communication patterns in VANETs," *IEEE Commun. Mag.*, vol. 46, no. 11, pp. 119–125, November 2008.

[5] M. Verma and D. Huang, "SeGCom: secure group communication in VANETs," in *Proc. IEEE Intl. Conf. on Consumer Comm. and Networking* (CCNC), Las Vegas, NV, USA, January 2009, pp. 1–5.

[6] S. Mittra, "Iolus: A framework for scalable secure multicasting," vol. 27, no. 4, pp. 277–288, 1997.

[7] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Comput. Surv.*, vol. 35, pp. 309–329, September 2003.

[8] C. K. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Networking*, vol. 8, no. 1, pp. 16–30, February 2000.

[9] D. Micciancio and S. Panjwani, "Optimal communication complexity of generic multicast key distribution," *IEEE/ACM Trans. Networking*, vol. 16, August 2008.

[10] A. Ballardie, "Scalable multicast key distribution," no. RFC 1949, May 1996, the Internet Engineering Task Force (IETF).

[11] J. Lin, K. Huang, F. Lai, and H. Lee, "Secure and efficient group key management with shared key derivation," *Computer Standards & Interfaces*, vol. 31, no. 1, pp. 192–208, 2009.

[12] P. Lee, J. Lui, and D. Yau, "Distributed collaborative key agreement and authentication protocols for dynamic peer groups," *IEEE/ACM Trans. Networking*, vol. 14, no. 2, pp. 263–276, April 2006.

[13] X. Li, Y. Yang, M. Gouda, and S. Lam, "Batch rekeying for secure group communications," in *ACM Proc. Intl. Conference on World Wide Web (*WWW*)*. Hong Kong, Hong Kong: ACM, May 2001, pp. 525–534.

[14] B. Briscoe, "MARKS: Zero side effect multicast key management using arbitrarily revealed key sequences," in *Networked Group Communication*, ser. Lecture Notes in Computer Science, L. Rizzo and S. Fdida, Eds. Springer Berlin / Heidelberg, 1999, vol. 1736, pp. 301–320.

[15] S. Busanelli, G. Ferrari, and L. Veltri, "Short-lived key management for secure communications in VANETs," in *Intl. Conference on ITS Telecommunications (*ITST*)*. Saint Petersburg, Russia: IEEE, August 2011, pp. 613–618.

[16] The Contiki Operating System. [Online]. Available: http://www.contiki-os.org

[17] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig, "Securing Communication in 6LoWPAN with Compressed IPsec," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2011)*, Barcelona, Spain, June 2011.