ARTICLE IN PRESS

Ad Hoc Networks xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

A novel batch-based group key management protocol applied to the Internet of Things

Luca Veltri [a], Simone Cirani [a,*], Stefano Busanelli [b], Gianluigi Ferrari [a]

[a] Department of Information Engineering, University of Parma, Italy
[b] Guglielmo Srl, Pilastro di Langhirano, Parma, Italy

ARTICLE INFO

Article history:
Received 15 October 2012
Received in revised form 28 March 2013
Accepted 20 May 2013
Available online xxxx

Keywords:
Internet of Things
Group key management
Security and privacy
Data aggregation

ABSTRACT

Many applications for ad hoc networks are based on a point-to-multipoint (multicast) communication paradigm, where a single source sends common data to many receivers, or, inversely, on a multipoint-to-point communication paradigm, where multiple sources send data to a single receiver. In such scenarios, communication can be secured by adopting a common secret key, denoted as "group key", shared by multiple communication endpoints. In this work, we propose a novel centralized approach to efficiently distribute and manage a group key in generic ad hoc networks and Internet of Things, while reducing the computational overhead and network traffic due to group membership changes caused by users' joins and leaves. In particular, the proposed protocol takes advantage of two possible leave strategies: (i) at a pre-determined time selected when the user joins the group or (ii) at an unpredictable time, as in the case of membership revocation. The proposed protocol is applied to two following relevant scenarios: (i) secure data aggregation in Internet of Things (IoT) and (ii) Vehicle-to-Vehicle (V2V) communications in Vehicular Ad hoc Networks (VANETs).

1. 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 [4], and Internet of Things (IoT), 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

* Corresponding author.
E-mail addresses: luca.veltri@unipr.it (L. Veltri), simone.cirani@unipr.it (S. Cirani), stefano.busanelli@guglielmo.biz (S. Busanelli), gianluigi.ferrari@unipr.it (G. Ferrari).

[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.

1570-8705/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2013.05.009

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 signatures), 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*.

Under the assumption of using security primitives unbreakable for an attacker with limited computational power, the main issue in group communications consists in distributing such secret group key to all the legitimated users and updating it at any group membership change. This problem is known as *Group Key Distribution* (GKD) and it can be tackled by following two different models [10]: (i) Broadcast Encryption (BE) [11,12], which assumes that current data be decipherable independently of past transmissions (the receivers are *stateless*) and (ii) Multicast Key Distribution (MKD), which allows the users to maintain state of the past cryptography material [13] (*stateful*). There are two main categories of MKD protocols: centralized [14] or distributed [15]. 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; and (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.

In a centralized MKD protocol, among the different methods to achieve secure communications in a group of n members, one of the simplest consists in having: (i) a group key, shared by group members only and changed every time a user joins or leave the group [16] and (ii) n individual long-term keys shared (pairwise) between the KDC and every group member. A message sent by a member to the whole group is encrypted with the group key, so that only the remaining members can decrypt it. Instead, the individual keys are used for securing unicast communications and for reeking. The management of the group keys has a cost, in terms of number message exchanges, that varies according to the protocol used to update the group key to all members (rekeying). In the simplest case, the KDC separately sends the new group key, encrypted with the member's long-term key, to all the group number, thus determining a number of exchanged messages proportional to n. The overall number of communications also depends on the average number of rekeying. In some protocols, the group key is refreshed suddenly after any join or leave events, in these cases the number of rekeying is directly proportional to the number of change of membership events. In other cases, the group key is refreshed programmatically according to a slotted schedule. These protocols have a constant number of rekeying operations, but they reduce the freedom of the nodes, and make impossible to perform instant evictions of malicious or dangerous users. Besides this classification, it is possible to define hybrid protocols, where join and leaves are performed programmatically, while evictions are performed immediately.

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 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 [16] that improves efficiency without posing security threats. The protocol also provides proper mechanisms to deal with unpredictable leave events and to resist against collusive attacks.

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 2 reviews related works. In Section 3, a new technique for solving the problem of distributing group-shared secret keys for ad hoc communications is proposed. In Section 4, the performance and the robustness of the proposed technique are assessed and compared with state-of-the-art protocols. In Section 5, the new technique is applied to a realistic application scenario for the Internet of Things and a particular application for Vehicular Ad hoc NETworks (VANETs). Finally, concluding remarks are given in Section 6.

2. Related works

In multicast group communications, a proper MKD protocol is required for generating and distributing a secret group key that 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.

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009

More precisely, the group key should be changed after every join and leave through a suitable *rekeying* operation, so that a former group member has no access to current communications and a new member has no access to previous communications [8]. These requirements can be expressed by introducing the concepts of *forward secrecy* and *backward secrecy* [9]. According to the former, non-members should not be able to obtain the group key at any instant based only on the information obtained at or before that instant. A more strict requirement, is the concept of *backward secrecy*, according with the group key at any instant should not be computable by non-members even after that instant (in other words, the new comers cannot compute past group keys). Moreover, group communication should be resistant to collusion attacks, in which (past or current) member of the group exchange information "out-of-band" in order to illicitly have access to information.

Join and leave operations can happen anytime (in an asynchronous and dynamic fashion), or alternatively, they can be synchronized at specific instants (in a slotted manner). In the second case, a number of join/leave operations should be jointly managed in the same temporal slot and for this reason these mechanisms are also referred as "batch" methods. In this work, we focus on this kind of mechanisms, since they significantly reduce the complexity (quantified in terms of number of exchanged messages) and they fit well the characteristics of realistic services, such as online game, where the join/leave operations have a daily or an hourly granularity.

In Section 1, MKD protocols have been classified in two main categories: centralized or distributed. While centralized key distribution protocols rely on a centralized key server (KDC) to efficiently distribute the group key, in distributed key agreement protocols there is no centralized server and the key is generated in a shared and contributory fashion by the member of the group, usually named peers. Such distributed protocols are often based on the multi-party extension of the well-known Diffie–Hellman key agreement protocol. For example Kim et al. [17] propose a Tree-based Group Diffie–Hellman protocol (TGDH) where each member maintains a set of keys, which are arranged in a hierarchical binary tree. A secret key and a blinded (or public) key are associated to every tree node. The secret key of a non-leaf node can be generated by the secret key of one child node and the blinded key of the other child node, similar to the classical two-party Diffie–Hellman protocol. Each group member is associated to a leaf node and the corresponding secret and blinded keys, that can be used to generate all secret keys along the path toward the root node secret key that, hence, can be generated by all nodes and corresponds to the group key. The TGDH protocol has been further optimized in [15]. However, distributed protocols need a larger amount of exchanged messages and operations, and they typically require full peer-to-peer communications, thus leading to a greater complexity than the more used centralized protocols. For these reasons centralized protocols are often preferred and used instead.

As described in Section 1, 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 [18], 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 [19], 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, the LKH scheme [8], to our slotted protocol.

3. New group key management protocol

In this section, a new group key distribution protocol is presented. The proposed protocol 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 summarized in Section 3.1, and detailed in Sections 3.2 and 3.3.

3.1. 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

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009

sections, this requires a suitable group key distribution protocol, 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; and (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 and (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 [18], 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 $K2$:

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

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 Δt_i (with $i = 0, 1, 2, \ldots, N$); they are predetermined 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 subkeys $K1_i$ 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.

3.2. 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 Δt_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 Δt equal to standard time units, such as a minute, a second, a month, etc. For each time interval Δt_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 Δt_a to time slot Δt_b: he/she will receive the subset of keys $S_X = \{K_i$, 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 Δt_h of member y that negotiated with KDC a membership from time slot Δt_a to time slot Δt_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 Δt_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 protocol is proposed.

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

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

where K_i, $K1_i$, and $K2$ are fixed or variable-length bit strings, and $f(\cdot)$ is a cryptographic one-way function that returns a bit string of length equal to or greater than K_i. If $f(\cdot)$ returns a bit string of length greater than K_i, a truncation can be applied. A cryptographic hash function $H()$ (for example SHA-1 [20] or MD5 [21]) can be used in place of $f(\cdot)$ as follows:

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

The subkey $K1_i$ 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 $0 \leqslant h \leqslant D$ and $0 \leqslant j \leqslant 2^h - 1$. 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:

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

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:

$$f_0(x) \doteq f(x)$$
$$f_1(x) \doteq f(x + 1)$$

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 subgraph 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 $(0,0)$.

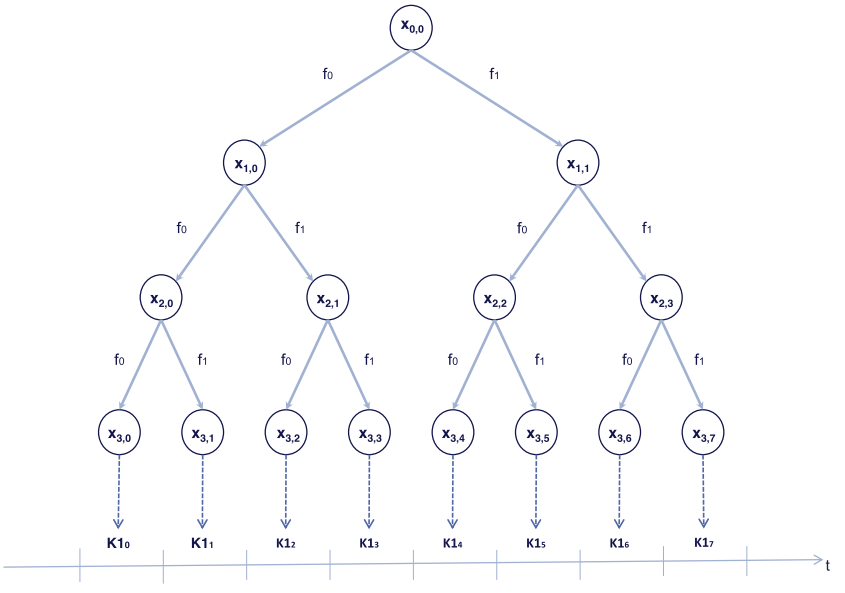

Fig. 1. Deriving all $K1$ subkeys by applying functions f_0 and f_1.

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009

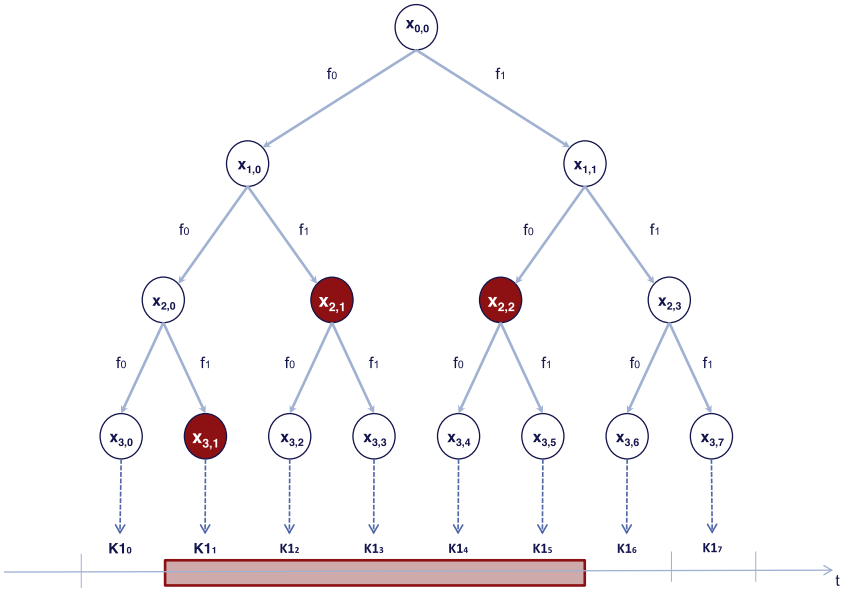

Fig. 2. Achieving backward and forward secrecy.

Given such a binary tree, we define the subkey $K1_i$ 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 $x_{0,0}$. Fig. 1 shows the $K1$ subkeys derivation process described above. At the same time, starting from the value $x_{h,i}$ of node (h,i) it is possible to obtain all subkey values in the interval from $2^{D-h} \cdot i$ to $2^{D-h} \cdot (i+1) - 1$ included, that is all subkeys 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 subkeys from $K1_0$ to $K1_{N-1}$. Fig. 2 shows how to obtain backward and forward secrecy by distributing the minimum set of values $x_{h,i}$ that cover the time period of a member's subscription.

This property can be used by the KDC to distribute the $K1_i$ subkeys 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 subkeys 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 subkey $K2$. 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.

3.3. 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 protocol 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 time intervals that might last beyond the one covered by a single tree (i.e., more than n time slots). The basic idea is to instantiate as many trees as required in order to manage a time interval of arbitrary length. This extension makes it possible to manage time intervals of any length with just a slightly increased computational effort and memory consumption.

Time is split into intervals I_k of length ΔT and periods Π_i of length $n \cdot \Delta T$. Each period Π_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, $h(\cdot)$ is a "*one-way*" function (i.e. hashing function), and $g(\cdot)$ is a "*blinding*" function (i.e. XOR function). For instance, possible choices for $h(\cdot)$ and $g(\cdot)$ are

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

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

$$s_0 = s \oplus H(iv)$$
$$s_1 = s \oplus H(H(iv)) = s \oplus H^2(iv)$$
$$\dots$$
$$s_i = s \oplus H^{i+1}(iv)$$

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

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

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 $a_0, a_1, \dots a_{h-1}$ are the h bits of the binary representation of I.

This mechanism makes it possible to extend the functioning of the protocol to cover an unlimited time interval without extra memory requirements as keys can be computed on the fly with no particular computational effort since operations like hashing and XORing are very lightweight.

4. Comparison of different key distribution strategies

In this section, the performance of the proposed key distribution protocol is compared with current state-of-the-art solutions, considering different application scenarios. The performance of the protocol is 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.

Table 1 shows the performances of the protocol when a node joins the group, a node leaves the group, or a node is evicted from a group of size n. The table shows the number of involved receivers and the amount of cryptographic material (values of the binary tree nodes to derive any K1 in the group subscription interval and K2 sub-keys) that each node must receive.

Note that, only in case of node eviction, the new K2 sub-key must be sent to all other group members. The total number of messages (M) needed to update K2 depends on the distribution strategy. A naive approach could be to send a separate message to each of the $n - 1$ members still being part of the group, but this method is clearly inefficient and can be considered a worst-case example. In order to optimize network traffic, we superimpose the LKH "*user-oriented rekeying*" strategy, which offers the best performance on the client side, but increases the computational effort of the KDC. We found this solution to best adapt to IoT scenarios, where devices typically offer little computational power and must minimize energy consumption. In this case, $M = (d - 1)(h - 1)$, where d is the LKH tree degree, and h is the length of the longest directed path of the LKH tree.

Tables 2–4 compare the performances of the LKH and MARKS protocols with our key distribution protocol, in case of join, leave, and eviction events, respectively. Note that, the super-imposed LKH strategy is necessary only if nodes are evicted, since the K1 sub-key alone guarantees forward secrecy when nodes leave the group gracefully.

As already said in Section 2, 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 comparison between the new protocol, MARKS, and LKH shows that the former achieves optimal performance, in terms of the metrics taken into account, for join events, predictable leave events, and unpredictable leave events. In the case of join and predictable leave events, the new protocol requires the same number of keys to be distributed and messages to be sent as MARKS.

The new approach considers also the case of node eviction, in order to provide a thorough description of all possible events that might occur in a group's lifecycle. Node eviction is not addressed by MARKS, while the proposed protocol manages this kind of events with the same performance as LKH both for the number of keys to be distributed and the number of messages to be sent.

Communication between KDC and group members is required only when nodes are evicted, thus communication overhead is kept to a minimum, thus ensuring optimal consumption of processing and network resources.

5. Key distribution in IoT scenarios

5.1. An application for sensor data aggregation

As aforementioned in Section 1, the scheme proposed in this work can be applied to 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. Encryption can be adopted not only to achieve

Table 1
Performance of the key distribution protocol evaluated in terms of number of keys to distribute and messages to be sent.

Event	Receivers	$x_{h,i}$ Values per node	K2 sub-keys per node	Total messages
JOIN	1	$\leqslant 2 \cdot (\log_2(N) - 1)$	1	1
LEAVE	0	0	0	0
EVICTION	$n - 1$	0	1	M

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009

Table 2
Comparison with LKH and MARKS in case of node joining group of size n.

Event	keys to distribute	messages
LKH	$h(h+1)/2-1$	h
MARKS	$\leqslant 2\cdot(\log_2(N)-1)$	1
ours	$\leqslant 2\cdot(\log_2(N)-1)$	1

Table 3
Comparison with LKH and MARKS in case of node leaving group of size n.

Event	Keys to distribute	Messages
LKH	$(d-1)h(h-1)/2$	$(d-1)(h-1)$
MARKS	0	0
Ours	0	0

Table 4
Comparison with LKH and MARKS in case of node being evicted from group of size n.

Event	Keys to distribute	Messages
LKH	$(d-1)h(h-1)/2$	$(d-1)(h-1)$
MARKS	Undefined	Undefined
Ours	$(d-1)h(h-1)/2+1$	$(d-1)(h-1)$

confidentiality, but also to verify the authenticity and integrity of messages.

Typically, secure data aggregation mechanisms require nodes to perform the following operations:

1. at the transmitting node, prior to transmission, data are encrypted with some cryptographic function E,
2. at the receiving node, all received data packets are decrypted with the inverse cryptographic function $D = E^{-1}$ to get the original data,
3. data are aggregate with an aggregation function,
4. prior to transmission, aggregate data are encrypted through E and relayed to the next hop.

This process is iterated until data reach the destination node who is interested in receiving the result of aggregation, as shown in Fig. 3. Both symmetric and asymmetric cryptographic scheme can be applied.

5.1.1. Asymmetric and symmetric encryption-based secure data aggregation

Public key (asymmetric) cryptography requires the use of a public key and a private key. Public keys can be associated with the identity of a node by including them into a public certificate, signed by a Certification Authority (CA) that can be requested to verify the certificate. Public key cryptography requires the significant effort of deploying a Public Key Infrastructure (PKI). Moreover, asymmetric cryptography requires higher processing and long keys (at least 1024 bits for RSA [22]) to be used. Alternative public key cryptographic schemes, such as Elliptic Curve Cryptography (ECC) [23], might require shorter keys to be used in order to achieve the same security than RSA keys. However, because of these reasons, symmetric cryptography is preferred in terms of processing speed, computational effort, and size of transmitted messages.

Symmetric cryptography requires the same key K to be used by the sender and the receiver to perform the encryption and decryption functions. Typically, a security association should be instantiated between each and every pair of communicating nodes, summing up to a total of $n\cdot(n-1)$ keys to be present in a community of size n. If some trust relationship is present among a number of nodes, 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.

5.1.2. Homomorphic encryption schemes

The secure data aggregation procedure described above, requires a significant amount of work to be performed on the aggregating node since all incoming packets need to be decrypted and the outgoing packets needs to be encrypted. These operations could be avoided by applying homomorphic encryption schemes. Homomorphic encryption is a form of encryption which allows specific types of computations to be executed on ciphertext and obtain an encrypted result which is the ciphertext of the result of operations performed on the plaintext:

$$E\{a\oplus b\} = E\{a\}\otimes E\{b\}$$

An example of a homomorphic encryption is the RSA algorithm. Let's consider a modulus N and an exponent e. The encryption of a message m is given by $E\{m\} = m^e \mathrm{mod} N$. The homomorphic property is

$$E\{m_1\cdot m_2\} = (m_1\cdot m_2)^e \mathrm{mod}\ N = (m_1)^e \mathrm{mod}\ N\cdot(m_2)^e \mathrm{mod}\ N = E\{m_1\}\cdot E\{m_2\}$$

Other examples of homomorphic encryption schemes are the ElGamal cryptosystem [24] and the Pailler cryptosystem [25]. It is clear that homomorphic encryption could dramatically improve the performances of a secure data

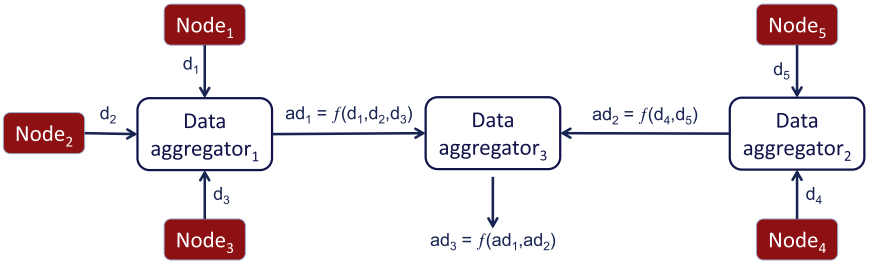

Fig. 3. In-network data aggregation.

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009

aggregation mechanism by allowing aggregating nodes to operate directly on encrypted data.

The examples cited above are all public key homomorphic encryption schemes. Therefore, they introduce the drawbacks of public key cryptography which we have outlined previously. A symmetric homomorphic encryption scheme could be a solution to achieve the benefits of efficient symmetric cryptography and avoid unnecessary computation on the aggregating node.

The Iterated Hill Cipher (IHC) is an additive symmetric homomorphic scheme, based on a modification of the original Hill Cipher in order to cope with its vulnerability to known-plaintext attacks. Riggio and Sicari employ modular addition as a symmetric key homomorphic function to perform secure aggregation in hybrid mesh/sensor networks [26]. Addition-based aggregation functions, such as sum and average, can be performed using either homomorphism.

5.1.3. Secure data aggregation with symmetric homomorphic encryption

In this work, we applied our group key distribution protocol to a particular application scenario where wireless sensors running the Contiki OS [27]. 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 [28]. 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.

5.2. An application for information dissemination in VANETs

5.2.1. The big picture

In this section, we focus on a particular type of Internet of Things scenario, already presented in [19], where the smart-objects are *vehicles*. More specifically, we consider the perspective of a service provider interested in offering a service (partially) based on VANET communications, adopting an approach similar to that proposed within the X-NETAD project [29]. The scenario of interest is shown in Fig. 4, where it is represented a single service provider disseminating information to disjoint clusters of authorized users through heterogeneous communications.

We consider a very simple Service Level Agreement (SLA) between the service provider and the users, which foresees three distinct categories of users: (i) Primary Users (PUs); (ii) Secondary Users (SUs); and (iii) Unauthorized Users (UUs). The PUs are privileged users that periodically retrieve the information of interest, from the service provider, through an Internet connection provided by a third-party Wide Area Network (WAN) such as a cellular network. As shown in Fig. 5, the PUs and the service provider can establish a unicast link that is relatively simple to secure by reusing standard cryptographic techniques and protocols (e.g., a TLS connection). The PUs propagate the information to a larger number of SUs, by means of multihop broadcast V2V communications, as shown in Fig. 5. The SUs obtain information from the PUs according to a push content distribution model (e.g., the PUs periodically disseminate it without any explicit query from the SUs), but they cannot receive the information directly from the service provider. Finally, the UUs are unauthorized users (e.g., users without a valid account) that must not have access to the information exchanged between PUs and SUs.

In practice, the distinction among the three different roles defined in the SLA is implemented through an access control service, e.g., a service restricting access to resources reserved to privileged entities. In particular, a *confidentiality service* is used to prevent UUs to have access to the reserved contents. The confidentiality service can be determined by encrypting the message in three different manners: (i) a group-shared secret key; (ii) a receiver-shared secret key (through symmetric cryptography); and (iii) a receiver public key (using asymmetric cryptography). In the first case, it is sufficient that the sender encrypts and multicasts only one copy of the message. Conversely, in the other cases, multiple copies of the message should be sent, each encrypted with a different receiver-shared or public key. For this reason, the problem of providing message confidentiality can be mapped into the problem of sharing a common group key between all the participants, under the assumption that a node can freely join or leave this group (for example, for commercial or administrative reasons). Coherently with these considerations, the information exchanged in a VANET will always be encrypted via symmetric cryptography and therefore, all the authorized users will share a common secret key (e.g., a group-shared key). It is important to observe that the choice of having a single group key shared across all the users, regardless of their geographical positions, allows to achieve a high degree of flexibility and scalability. In fact, this enable the local clusters of nodes (visible in Fig. 4) to split or merge together without the need of renegotiate a new group key.

Once the confidentiality service has been established, the distinction between PUs and SUs can be easily managed by the service provider in a centralized manner. Without loss of generality, it can be assumed that the subscriptions have limited durations and, hence, have to be periodically renewed. Moreover, since a user could issue

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009

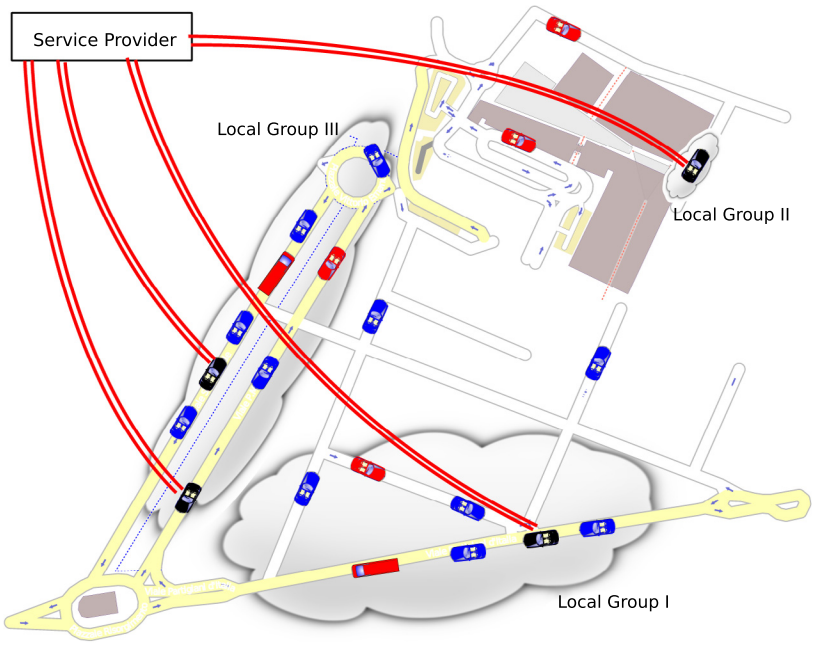

Fig. 4. General view of the data dissemination application.

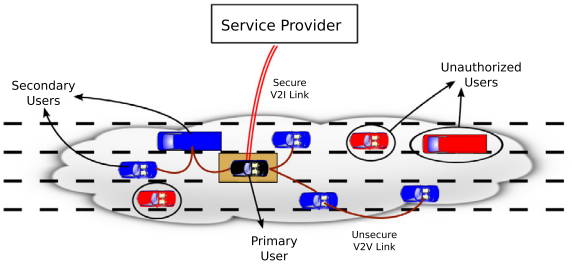

Fig. 5. A graphical representation of the data-flow of the considered application.

(or cancel) a new subscription at any time, its status could unpredictably change over time. It is straightforward to observe the needs of the confidentiality service are perfectly addressed by the scheme designed in Section 3.

5.2.2. Security architecture

Since the exchange of information between PUs and SUs is entirely based on V2V communications, the application requires the presence of dedicated (or pre-existing) VANETs. Regardless of the nature of the VANETs, we assume that the vehicles are equipped with OBUs compliant with the IEEE 802.11p standard [30] and adhering to the IEEE 1609 protocols suite. In particular, the OBUs implement all the security services defined in the IEEE 1609.2 standard. The application can be installed on the on-board computer of a vehicle or it can be executed on a smartphone or other commercially available equipments (e.g., GPS navigators). The application should be able to send and receive packets through the OBU by means of a local connection (e.g., Bluetooth or a wired communication bus).

Generally speaking, a VANET should provide several distinct security services, which can be summarized as follows [31].

- *Confidentiality*: keeping secret the information from UUs.
- *Access control*: restricting the access to reserved resources for PUs and SUs.
- *Integrity*: ensuring that the transmitted information cannot be modified by UUs.
- *Authentication*: certifying that the origin of a message is correctly identified.
- *Identification*: establishing the identity of a user.
- *Non-repudiation*: preventing the sender of a message from denying having created that message.

In the multihop communication scenario presented in Section 5.2.1, the vehicles route messages trough the entire network by broadcasting them to all neighbors within their coverage area. In order to protect messages against eavesdropping, modification, and/or fake message forging by unauthorized malicious nodes, message authentication and confidentiality should be properly implemented before sending messages to the neighbors.

As anticipated in Section 5.2.1, each vehicle supports the IEEE 1609.2 protocol suite. In particular, the IEEE 1609.2 standard defines the following security functionalities [32]:

- digital signatures using ECC [23], specifically Elliptic Curve Digital Signature Standard (ECDSA), by using the Fp curves defined by the NIST [33];
- asymmetric encryption with ECC, specifically Elliptic Curve Integrated Encryption Scheme (ECIES) [34,35];

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009

- purely symmetric scheme: authenticated encryption, specifically counter mode encryption, and CBC-MAC with AES (AES-CCM) [36].

Therefore, the IEEE 1609.2 standard defines primitives sufficient for providing data authentication, non-repudiation, data integrity, and confidentiality (under the assumption of having an encryption key shared across all the authorized users). However, the IEEE 1609.2 standard does not address several critical questions, namely: (i) privacy; (ii) key management; and (iii) identity management.

In order to address these issues, several security architectures have been proposed in the literature. In this work, we have considered a general architecture largely based on that introduced in [37]. In particular, the node architecture is shown in Fig. 6. Coherently with the main goal of this work, we will not detail the characteristics of the security infrastructure, but we will only sketch it.

First of all, a PKI, constituted by several CAs, is established. The vehicles communicate with the CAs through either cellular-based or RSU-based communications. Each vehicle has a unique IDentification (ID), a pair of long-term private–public keys, and a long-term certificate issued by a CA upon node registration. Actually, the long-term credentials are not directly used, but they are needed to issue second-level short-term credentials, denoted as "pseudonyms", which are actually used for securing V2V communications. The goal of using pseudonyms is twofold: (i) to reduce the risk of compromising the long-term credentials; (ii) to increase the privacy of the vehicle. However, it is important to remark that since short-term certificates need to be issued from CAs, then pseudonyms can be used only if Internet accesses are frequently available to the vehicles. In the case of the considered application, the PUs continuously have an Internet connection. To summarize, with respect to a legacy IEEE 1609.2 system, the proposed architecture provides (i) a higher level of privacy and (ii) key and identity management systems.

It should be now addressed the problem of deriving and distributing a common group key across all the authorized users. From a logical point of view, the problem of generating a common group key is the same addressed in detail in Section 3. In fact, the service provider is the unique entity with the right to authorized the network users, therefore, it can act as a full-fledged KDC reusing the technique described in Section 3 for managing the user subscriptions and for generating the new group keys. Instead, the distribution of the new group keys is hindered by the lack of direct links between the SUs and the KDC. In order to solve this problem we have adopted a two phases strategy. In the first phase, the KDC distributes the new group keys to the PUs by using the unicast links existent between them, that can be easily secured by using standard cryptographic tools (as described in Section 5.2.1). In the second phase, the PUs distribute the new group keys to the SUs belonging to their local clusters. The distribution is carried out by establishing a temporary unicast link between the PU and each SU. The group keys can be encrypted, either with symmetric or asymmetric cryptographic methods, by using the encryption primitives provided by the IEEE 1609.2 stack and the cryptographic materials associated to the pseudonyms.

It is now possible to define the operations performed by a PU to broadcast an encrypted packet to the SUs of its local cluster. We remark that the symbol k denotes the common group key.

1. To generate two sub-keys k_i and k_e from the common secret key k.

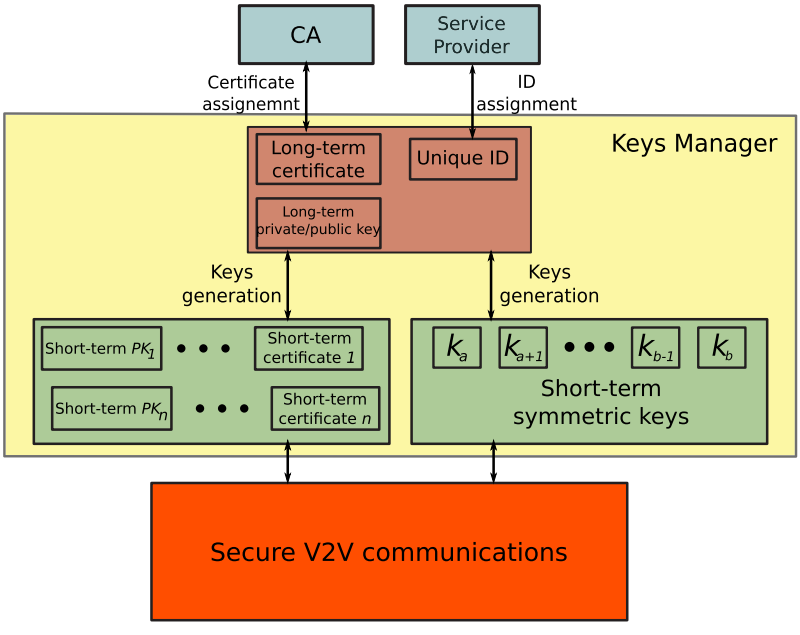

Fig. 6. Security architecture of the considered application.

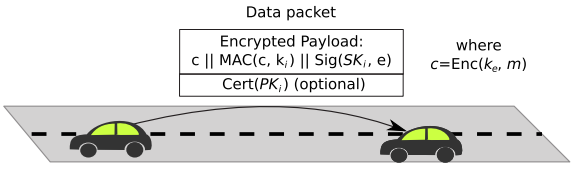

Fig. 7. Structure of the packet with emphasis on its encrypted payload.

2. To encrypt the message (denoted as m) with k_e: $c = \mathrm{Enc}(k_e, m)$.
3. To compute the Message Authentication Code (MAC) of c with the key k_i: $\mathrm{MAC}(k_i, m)$.
4. To compute the signature of c with the private key Sk_i: $\mathrm{Sig}(Sk_i, c)$.
5. To compose the packet as follows: $c \| \mathrm{MAC}(ki, c) \| \mathrm{Sig}(Sk_i, c)$
6. If required, to attach the certificate Cert (PK_i) and, finally, to send the message.

The structure of the packet built according to these operations is shown in Fig. 7.

Once the encrypted message has been correctly received, a SU should perform the following operations.

1. To obtain the public key PK_i from Cert (PK_i) (if it is attached).
2. To generate two sub-keys k_i and k_e from the common secret key k.
3. To verify the integrity of the message, by computing the MAC of c with the key k_i, and compare it with the received MAC.
4. To verify the signature of c with the public key PK_i (if known). If user does not know the public key it can also decide not to verify the signature at all.
5. To finally decrypt the message c with k_e, after having verified the authenticity and the integrity of the packet: $m = \mathrm{Dec}(k_e, c)$.

Finally, it is worth to be mentioned that the described system has been implemented and validated on a platform based on Android smartphones equipped with GPS, IEEE 802.11 interface, and 3G interface. According to the preliminary results, the system has worked as expected exhibiting a performance level coherent with the data presented in Section 4. Unfortunately, because of the small number of available devices, we were unable to perform experiments on large scale scenario, e.g. thousands of nodes.

6. Conclusions

In this paper, we have presented an innovative protocol 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 pre-determined leave event can be easily handled. Only in case of key revocation events explicit communication between KDC and group member is required, allowing our 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.

Acknowledgments

The work of Luca Veltri, Simone Cirani, and Gianluigi Ferrari is funded by the European Communitys 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 (RFC 1112), The Internet Engineering Task Force (IETF).
[2] T. Turletti, C. Huitema, Videoconferencing on the Internet, IEEE/ACM Trans. Netw. 4 (3) (1996) 340–351.
[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Comput. Netw. 38 (4) (2002) 393–422.
[4] E. Schoch, F. Kargl, M. Weber, T. Leinmuller, Communication patterns in VANETs, IEEE Commun. Mag. 46 (11) (2008) 119–125.
[5] M. Verma, D. Huang, SeGCom: secure group communication in VANETs, in: Proc. IEEE Intl. Conf. on Consumer Comm. and Networking (CCNC), Las Vegas, NV, USA, 2009, pp. 1–5.
[6] S. Mittra, Iolus: A framework for scalable secure multicasting, in: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication, 1997.
[7] S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Comput. Surv. 35 (2003) 309–329.
[8] C.K. Wong, M. Gouda, S. Lam, Secure group communications using key graphs, IEEE/ACM Trans. Netw. 8 (1) (2000) 16–30.
[9] D. Micciancio, S. Panjwani, Optimal communication complexity of generic multicast key distribution, IEEE/ACM Trans. Netw. 16, 2008.
[10] M. Gerla, L. Kleinrock, Vehicular networks and the future of the mobile Internet, Comput. Netw. 55 (2) (2011) 457–469.
[11] S. Berkovits, How to broadcast a secret, in: Proc. of the Intl. Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT), Springer-Verlag, Brighton, UK, 1991, pp. 535–541.
[12] D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: J. Kilian (Ed.), Advances in Cryptology (CRYPTO), Lecture Notes in Computer Science, vol. 2139, Springer, Berlin/Heidelberg, 2001, pp. 41–62.
[13] A. Ballardie, Scalable Multicast Key Distribution (RFC 1949), the Internet Engineering Task Force (IETF).
[14] J. Lin, K. Huang, F. Lai, H. Lee, Secure and efficient group key management with shared key derivation, Comput. Stand. Inter. 31 (1) (2009) 192–208.
[15] P. Lee, J. Lui, D. Yau, Distributed collaborative key agreement and authentication protocols for dynamic peer groups, IEEE/ACM Trans. Netw. 14 (2) (2006) 263–276.

[16] X. Li, Y. Yang, M. Gouda, S. Lam, Batch rekeying for secure group communications, in: ACM Proc. Intl. Conference on World Wide Web (WWW), ACM, Hong Kong, Hong Kong, 2001, pp. 525–534.
[17] Y. Kim, A. Perrig, G. Tsudik, Tree-based group key agreement, ACM Trans. Inf. Syst. Secur. 7 (2004) 60–96.
[18] B. Briscoe, MARKS: zero side effect multicast key management using arbitrarily revealed key sequences, in: L. Rizzo, S. Fdida (Eds.), Networked Group Communication, Lecture Notes in Computer Science, vol. 1736, Springer, Berlin/Heidelberg, 1999, pp. 301–320.
[19] S. Busanelli, G. Ferrari, L. Veltri, Short-lived key management for secure communications in VANETs, in: Intl. Conference on ITS Telecommunications (ITST), IEEE, Saint Petersburg, Russia, 2011, pp. 613–618.
[20] Federal Information Processing Standards Publication, FIPS 180-3. Secure Hash Standard (SHS) (2008). <http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html>.
[21] R. Rivest, RFC 1321: The MD5 Message-Digest Algorithm, the Internet Engineering Task Force (IETF).
[22] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21 (1978) 120–126.
[23] N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation 48 (177) (1987) 203–209.
[24] T. El Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in: Proceedings of CRYPTO 84 on Advances in Cryptology, Springer-Verlag New York, Inc., New York, NY, USA, 1985, pp. 10–18. <http://dl.acm.org/citation.cfm?id=19478.19480>.
[25] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: J. Stern (Ed.), Advances in Cryptology – EUROCRYPT +99, Lecture Notes in Computer Science, vol. 1592, Springer, Berlin/ Heidelberg, 1999. 10.1007/3-540-48910-X_16. <http://dx.doi.org/10.1007/3-540-48910-X_16>.
[26] R. Riggio, S. Sicari, Secure aggregation in hybrid mesh/sensor networks, in: ICUMT'09. International Conference on Ultra Modern Telecommunications & Workshops, 2009, IEEE, 2009, pp. 1–6.
[27] The Contiki Operating System. <http://www.contiki-os.org>.
[28] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, 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, 2011.
[29] Eureka Project 6252 X-NETAD. <http://www.eurekanetwork.org/project/-/id/6252>.
[30] IEEE Standard for Information Technology – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments, IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007), 2010, pp. 1–51.
[31] A. Weimerskirch, J.J. Haas, Y.-C. Hu, K.P. Laberteaux, Data security in vehicular communication networks, in: H. Hartenstein, K. Laberteaux (Eds.), VANET Vehicular Applications and Inter-Networking Technologies, Wiley Press, 2010, pp. 299–365 (Chapter 9).
[32] Insitute of Electrical and Electronics Engineers, IEEE 1609.2-2006. IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) – Security Services for Applications and Managem.
[33] Federal Information Processing Standards Publication, FIPS 186-3. Digital Signature Standard (DSS), 2009. <http://csrc.nist.gov/>.
[34] Insitute of Electrical and Electronics Engineers, IEEE 1363-2000. IEEE Standard Specifications for Password-Based Public-Key Cryptographic Techniques, 2000.
[35] International Organization for Standardization, ISO/IEC 18033-2:2006 Information Technology – Security Techniques – Encryption Algorithms – Part 2: Asymmetric Ciphers, 2006.
[36] National Institute of Standards and Technology (NIST), NIST Special Publication 800-38c: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, March 2004. <http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf>.
[37] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, J. Hubaux, Secure vehicular communication systems: design and architecture, IEEE Commun. Mag. 46 (11) (2008) 100–109.

Luca Veltri received his Laurea degree in Telecommunication Engineering and his Ph.D. in Communication and Computer Science from University of Rome "La Sapienza" in 1994 and 1999, respectively. Since 2002 he is assistant professor at University of Parma (Italy), where is currently working and where he teaches classes on Telecommunication Networks, and Network Security. His main research fields are: P2P systems, Future Internet, and Network Security.

Simone Cirani was born in Asola (MN) on October 1982. He received his Dr. Ing. (Laurea) degree in Information Engineering "cum laude" from the University of Parma, Italy, in 2007 with a thesis entitled "Development of a DHT-based peer-to-peer architecture for the implementation of a distributed SIP Location Service". In 2011, he received his Ph.D. at the Department of Information Engineering of the same university.His research interests are Peer-to-peer networks, Security in Internet of Things, Pervasive computing, and Social media integration and analysis.

Stefano Busanelli was born in Castelnovo né Monti (RE), on September 1982. He received the "Laurea" degree (3-year program, equivalent to a Bachelor) in Telecommunications Engineering summa cum laude on December 2004, discussing a thesis with title "Signaling platform for peer to peer applications" (in Italian). On December 2007, he received the "Laurea" degree (3 + 2 year program, equivalent to a Master) in Telecommunications Engineering "summa cum laude," discussing a thesis with title "Markov chains-based performance analysis of wireless sensor network with multihop communications" (in Italian). In March 2011, he got his Ph.D. in Information Technology from the University of Parma, discussing a thesis with title "Efficient Multihop Wireless Communications in VANETs."

From June 2008 to October 2008, he was an intern at Thales Communications (Colombes, France), collaborating with Dr. Christophe Le Martret and Dr. Isabelle Icart. His research has been focused on performance analysis of distributed Space–Time Block Codes and design of cooperative MAC protocols for ad hoc wireless network.

Since March 2011, he is a Post-doc researcher at the Information Engineering Department (DII) of the University of Parma, under the supervision of Prof. Gianluigi Ferrari, working on wireless communications in vehicular ad hoc networks. His activity carried out under the one-year project "Cross-Network Effective Traffic Alerts Dissemination" (X-NETAD, Eureka Label E! 6252), sponsored by the Ministry of Foreign Affairs (Italy) and The Israeli Industry Center for R&D (Israel) under the "Israel-Italy Joint Innovation Program for Industrial, Scientific and Technological Cooperation in R&D." He is a member of the Wireless Ad hoc and Sensor Networks (WASN) laboratory. Dr. Busanelli has taken part into research projects in collaboration with a few research organizations and private companies. He is also a frequent reviewer for some international conferences and international journals. He got an "Award for outstanding contribution" at 11th International Conference on Telecommunications for Intelligent Transport Systems (ITST-2011), Saint Petersburg, Russia. He also won the first prize award, together with the WASNLab team, at the first Body Sensor Network (BSN) Contest, organized in conjunction with the 2011 Body Sensor Networks (BSN '11) conference, Dallas, TX, USA, May 2011.

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009

Gianluigi Ferrari was born in Parma, Italy, on November 13, 1974. He received the "Laurea" degree (5-year program) in Electrical Engineering "summa cum laude" from the University of Parma, Parma, Italy in October 1998. He received the Ph.D. degree in "Information Technologies" from the Department of Information Engineering (DII) of the University of Parma in January 2002. From July 2000 to December 2001 he was a Visiting Scholar at the Communication Sciences Institute, University of Southern California, Los Angeles, California, USA. Between February 2002 and August 2002, he was a Postdoc Student at the DII, University of Parma.

Between September 2002 and October 2010, he was a Research Professor at the DII of the University of Parma. Since November 2010, he has been an Associate Professor at the same department (he received the Italian nation-wide habilitation from Polytechnic of Milan in August 2010). During October 2002–February 2003, July-December 2003 and July–December 2004, he was a Research Associate at the Electrical and Computer Engineering Department of Carnegie Mellon University, Pittsburgh, PA, USA. Since September 2006 he has been the Coordinator of the Wireless Ad hoc and Sensor Networks (WASN) Laboratory. In Fall 2007 he visited, as a DUO-Thailand Fellow, the King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand. Between July 2010 and October 2010, he was a Visiting Researcher, within a "Brains (Back) to Brussels" (B2B) program, at the OPERA Department of the Université Libre de Bruxelles (ULB), Belgium.

As of July 2012, he has (published or in press) more than 180 papers in international journals and international/national conference proceedings, 17 book chapters, nine books (six in English and two in Italian), and nine patents (Italian and international). He is a co-recipient of a Best Student Paper Award at the 2006 International Workshop on Wireless Ad Hoc Networks (IWWAN'06), New York, NY, USA, June 2006; a Best Paper Award at the Second International Conference on Emerging Network Intelligence (EMERGING 2010), Florence, Italy, October 2010; an Award for Outstanding Contribution to the 11th International Conference on ITS Telecommunications (ITST-2011), Saint Petersburg, Russia, August 2011; the Best Paper Award at the 1st International Conference on Sensor Networks (SENSORNETS 2012), Rome, Italy, February 2012; the best paper award at EvoComNet 2013. The WASNLab team won the first Body Sensor Network (BSN) contest, held in conjunction with BSN 2011.

He is a frequent reviewer for many international conferences and journals and acts as a technical program committe (TPC) member for many international conferences. He has also co-organized a few international workshops and is currently member of the Editorial boards of a few international journals. He has collaborated in several research projects, among which some were funded by the (Italian) Ministry of Instruction, University, and Research (MIUR), the (US) Army Research Office (ARO), Regione Emilia-Romagna, the (Italian) Ministry of Foreign Affairs (MAE), the European Commission (FP7 program), and several private companies, both Italian and international. To date, the amount of funding brought to the Department of Information Engineering is over 1 MEuro.

Please cite this article in press as: L. Veltri et al., A novel batch-based group key management protocol applied to the Internet of Things, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.05.009