Fault-Tolerant and Scalable Key Management Protocol for IoT-Based Collaborative Groups

Mohammed Riyadh Abdmeziem(✉) and François Charoy

Université de Lorraine Inria-CNRS-LORIA, Nancy, France
{mohammed-riyadh.abdmeziem,francois.charoy}@loria.fr

Abstract. Securing collaborative applications relies heavily on the underlying group key management protocols. Designing these protocols is challenging, especially in the context of the Internet of Things (IoT). Indeed, the presence of heterogeneous and dynamic members within the collaborative groups usually involves resource constrained entities, which require energy-aware protocols to manage frequent arrivals and departures of members. Moreover, both fault tolerance and scalability are sought for sensitive and large collaborative groups. To address these challenges, we propose to enhance our previously proposed protocol (i.e. DBGK) with polynomial computations. In fact, our contribution in this paper, allows additional controllers to be included with no impact on storage cost regarding constrained members. To assess our protocol called DsBGK, we conducted extensive simulations. Results confirmed that DsBGK achieves a better scalability and fault tolerance compared to DBGK. In addition, energy consumption induced by group key rekeying has been reduced.

Keywords: Collaborative applications · Internet of Things (IoT)
Security · Group key management · Polynomial computation · Contiki

1 Introduction

With the rise of the Internet of Things (IoT) and its integration in information systems, collaborative applications have taken a new dimension. Pervasive devices and objects are able to perceive our direct environment and act autonomously upon it to help users to reach their goals. Applications flourished in healthcare, transportation and military environments [4] that combine input from users and objects to reach goals in a collaborative way. In these domains, stakeholders would only accept these systems in their environment if they have strong guarantees on the security, privacy and integrity of the data they produce and share. The distributed nature of such systems and the requirement for encryption of data shared among participants lead to one of the most important challenges in such evolving environments: the management of cryptographic group keys [2,6,32].

X. Lin et al. (Eds.): SecureComm 2017 Workshops, LNICST 239, pp. 320–338, 2018.
https://doi.org/10.1007/978-3-319-78816-6_22

Group key management is challenging in this context. In fact, collaborative groups involve heterogeneous members with different requirements and resources capabilities [17]. This gap can hinder end-to-end communications. Indeed, constrained members with limited processing power and storage space can not run heavy cryptographic primitives [5]. Moreover, collaborative applications may present a high rate of leaving and joining members within tight time lapses, which makes the issue more difficult to handle. The scalability of these systems needs to be addressed bearing in mind the increasing number of entities taking part in the collaborative groups. Last, fault tolerance is at utmost importance especially for critical and sensitive applications (e.g. health related and military applications) [31].

We address this problematic of designing a secure and efficient protocol to establish shared group credentials for Peer-to Peer collaborative groups. These credentials will be used to ensure the required security properties such as data confidentiality, data integrity, and data authentication. The proposed protocol has to be energy aware allowing an implementation on constrained devices, which take part in the collaborative process. In addition, the protocol must be scalable, as well as tolerant to possible failures of the entity in charge of managing the group key.

To achieve this goal, we rely on our previously proposed group key management protocol called DBGK (Decentralized Batch-based Group Key) [3]. This protocol considers a network topology composed of several sub groups. Each sub group is managed by an area key management server, while the whole group is managed by a general group key management server. The established group key is composed of a long term key and short terms keys (called tickets), which are different for each time interval. Constrained members in terms of resources (e.g. connected objects) are only involved in the re-keying process if these latter have recently been active. In addition, keying materials are distributed to joining members based on their resources capabilities. Experiments showed that DBGK [3] is energy efficient and outperforms similar existing protocols in the literature.

Although efficient and secure, DBGK relies on key management servers to maintain the group key. Including additional servers to improve fault tolerance would impose a high storage overhead on constrained members. This makes DBGK inappropriate to be directly implemented in sensitive collaborative applications. In this paper, we propose a distributed extension for DBGK called DsBGK (Distributed Batch-based Group Key). In this extension, we keep the core functioning of DBGK, while significantly distributing the operations which were based on a central entity. We achieve this by integrating a polynomial based scheme inspired from [24,25]. In addition, we improve the efficiency of the original scheme to suit the constrained IoT environment. We conducted extensive experiments to assess the performances of DsBGK and compared the results with DBGK performances. The results showed that DsBGK provides an enhanced scalability and fault tolerance, as additional key management servers (controllers) can be included without impacting the storage overhead on constrained members. Furthermore, energy cost due to rekeying operations is

reduced compared to DBGK, which extends the life cycle of battery powered entities.

The remaining of the paper is organized as follows. In Sect. 2, we present a use case scenario to motivate our contribution. In Sect. 3, we discuss, in detail, existing solutions in the literature. For the sake of clarity, we summarize in Sect. 4, the required background. In Sect. 5, we present our network model, along with our assumptions and the used notations. In Sect. 6, we thoroughly present our approach before introducing and analyzing the experimental results in Sect. 7. Section 8 concludes the paper and sets our future direction.

2 Use Case Scenario: Personal Health Record (PHR)

A personal heath record [33] (Fig. 1) is a typical example of a document that can be accessed and edited by multiple participants, including medical sensors attached to patients. This is also an example of a document that contains highly private and sensitive information. To edit a medical record, some participants (e.g. medical staff) collaborate using unconstrained devices, such as Personal Computers (PC) and smartphones. However, sensors planted in or around the human body are considered as constrained since they have limited computing power and may operate on battery. These sensors can either communicate their sensed data to medical staff through the unconstrained entities (e.g. PC, smartphones) or directly edit patient's medical record. Medical staff can also control the sensors (trigger or stop the sensing of a particular physiological data), and add more sensors to the collaboration. New members can join or leave the collaboration around the medical record as the situation of the patient evolves. The different entities collaborate in a distributed way to maintain the medical

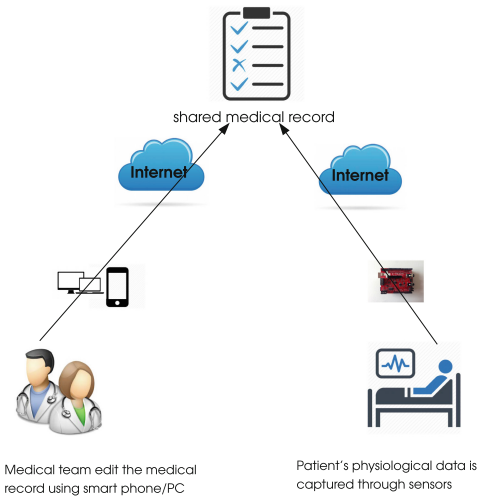

Fig. 1. Use case scenario

record. This latter can be replicated among different entities and the modifications can be executed on the different replicas, which need to be synchronized. This is important in order to avoid a single point of failure on the record management architecture. It is also important to control the entities that have access and can modify the record over time. This clearly highlights the importance of securing communications in such a hybrid and heterogeneous group of entities by efficiently managing the security credentials used to provide data authentication and data confidentiality. Personal Health Record (PHR) is a typical case of collaboration among health-care personal, insurers, caregivers, patients and sensors to maintain a document that reflects the patient status, health history and treatment. There is an obvious need to provide a decentralized, secure, safe, privacy preserving and scalable solution to share these documents among people and sensors (objects).

3 Related Work

In this section, we review the main categories under which group key management protocols are usually categorized [11,28], namely, the centralized, the decentralized, and the distributed categories.

Centralized protocols are based on an unconstrained central entity (i.e. Key Management Server (KMS)), which is responsible for generating, distributing, and updating the group key for the whole group. Authors in [15] introduced the Group Key Management Protocol (GKMP), which is based on a Group Key Packet (GKP). This latter encompasses a Group Traffic Encryption Key (GTEK) to secure data traffic, and a Group Key Encryption Key (GKEK) to secure transmissions related to rekeying operations. Following a leave event, the central entity broadcasts the new GKP to all remaining members creating a complexity of $O(n)$. This complexity makes GKMP not scalable with regards to dynamic and large groups. To reduce the impact of leave events, authors in [34] proposed an interval-based protocol, which generates the keying materials corresponding to the predicted period of time during which the members are expected to remain in the group. Doing so, following a leave event, no rekeying is required. However, this solution is not suited to dynamic groups with unexpected join and leave events, as predicting the leaving moment of members is neither realistic nor practical. In addition, constrained members which are part of the group for a long period of time might suffer from storage issues, as a large number of keying materials needs to be stored.

To further improve efficiency, several hierarchical based protocols have been proposed. Among them, the Logical Key Hierarchy (LKH) protocol [37], later improved by the One-way Function Tree protocol [7] are typical examples. The basic idea of these protocols is that the KMS shares pre-established credentials with subsets of the group. Following an event, the KMS relies on these credentials to target specific subgroups during the rekeying, thus, reducing the number of required rekeying messages (i.e. $OLog(n)$).

Thanks to their efficiency, polynomial based approaches are used to manage group keys in collaborative applications. In fact, polynomial based schemes

allow overcoming the storage cost related to multicast inter-group communications. Moreover, polynomial evaluation can be, under certain conditions, more efficient than encryption/decryption primitives. Polynomials have originally been included in threshold secret sharing schemes [30]. More recently, authors in [35,36] used polynomials to enable group members decrypting received messages. Doing so, the members are no longer required to store a secret key shared with each sender. Nevertheless, polynomials are usually generated and broadcasted by the KMS. To reduce this overhead on the KMS, authors in [25] propose a self-generation technique to generate the polynomials by the members of the group. In a nutshell, centralized protocols are characterized by their efficiency due to the use of symmetric primitives. Furthermore, these protocols do not require peer-to-peer communications during rekeying operations. However, the single point of failure and scalability issues constitute their main weaknesses.

Decentralized protocols consider the group divided into various areas, with an Area Key Management Server (AKMS) in charge of managing local events. This class of protocols is generally categorized into two sub categories [11]: *common Traffic Encryption Key (TEK) per area* [9,27], and *independent TEK per area* [22,25]. In the former category, a unique TEK is implemented for the various areas of the group. As a result, if an event happens, the whole group is affected by the rekeying. In the latter category, a different TEK is implemented for each area. As a result, the *1-affects-n* issue is attenuated, as rekeyings only affect specific areas. However, data transmitted across areas has to be translated at the border of each area. This classification of decentralized protocols can further be refined [10] by including *time-driven* rekeying subcategory [9,29] and *membership-driven* rekeying subcategory [8,27]. In membership-driven protocols, the group key is updated following each membership event, whereas, in time-driven protocols, the update of the group key is carried out at the end of a defined period of time without taking into consideration membership events. Consequently, the impact of frequent and consecutive events is limited. Nevertheless, ejected members are still able to access exchanged data up to the end of the interval. Likewise, a new member would have to temporize until the start of a new interval prior of being able to access exchanged data in the group.

Distributed protocols do not rely on any central entity. Instead, all members contribute in the management of the group key in a peer-to-peer way. Distributed protocols are usually based on the n-party version of the well known Diffie-Hellman protocol [18,19]. Hence, these protocols are highly reliable, as the group is free from any single point of failure. Nevertheless, distributed protocols involve a high number of exchanged messages during rekeying operations, in addition to an important computation cost due to the use of heavy asymmetric primitives.

To alleviate this cost, authors in [13] propose a probabilistic based protocol. Members of the group establish communication channels composed of sequences of adjacent members between which a key is shared. Indeed, members propagate the key, which is shared between the first adjacent members to the remaining members. This propagation is achieved using local keys. However, if no local key is found between two specific members, these members proceed with a pairing

attempt by exchanging a set of global keys generated from a pool of keys. In spite of its improved performances compared to deterministic protocols, this protocol suffers from a lack of connectivity. In fact, members could be disconnected from the group if several pairing attempts fail. To further mitigate the complexity of distributed protocols, authors in [12] introduce a protocol which proceeds within two phases. In the first phase, members of the group autonomously generate the group key using pre-defined seeds and hash functions. In the second phase, members synchronize their generated keys taking into account delays due to the loose synchronization of members clocks. Compared to other solutions based on DH primitives, one of the drawbacks of this protocol lies in the pre-sharing assumption of the seeds, which affects both its scalability and feasibility.

In this context, we address the issue of group key management for dynamic and heterogeneous collaborative groups. The originality and features of our approach are detailed through the remaining sections. But first, to ease the understanding of our contribution, we provide the reader with a broad overview of the protocols upon which our approach is built.

4 Background

4.1 DBGK [3]

DBGK considers the group divided into sub groups. Each sub-group is managed by an Area Key Management Server ($AKMS$). The time axis is split into several time slots. For each time slot, a different ticket (piece of data) is issued. The group Traffic Encryption Key (TEK) for slot i is computed using a one way function F as follows:

$$TEK_i = F(SK, T_i)$$

where SK is a long term key, and T_i is the ticket issued for slot i.

Once an object (or member, both terms are used indistinguishably) O_i wants to join the group, it initiates DBGK which goes through successive phases. The object sends a join request through an anycast message. Based on the object location, the nearest $AKMS$ handles the join. Let us assume that the $AKMS$ of area j is the nearest one. In case of a successful authentication, the object is initialized (through a secure channel) with a long term key (i.e. SK), and a shared key with its $AKMS$. Despite being a valid member of the group, the new member O_i is not yet able to derive the current TEK. Backward secrecy is therefore inherently ensured while no rekeying operation is required for the group. If O_i is involved in a message exchange (sending/receiving), it has to be able to encrypt and decrypt the messages. To do so, O_i has to compute the current TEK. Thus, O_i sends a request to $AKMS_j$ asking for a ticket corresponding to the current time slot. In order to reduce the amount of exchanges in case O_i is highly active, the object can request several tickets corresponding to multiple future intervals. The request contains information about the objects specifications, in particular, data regarding its storage capabilities and resources. Based on this data, and on the trust level of O_i (if the object has previously

been a member of the group), AKMS decides on the number of tickets to be granted to O_i.

When O_i leaves the network, forward secrecy has to be guaranteed to prevent the object from accessing future communications in the area. Two possible scenarios arise. In the first case, O_i leaves the network or is ejected with one or several valid tickets stored in its internal memory. In this case, $AKMS$ checks its AOL (Active Object List, which keeps track of the issued tickets) and sends a multicast notification to all the objects that have received the same tickets owned by the leaving member. The semantics of the notification is as follows. The tickets ranging from T_t to T_{t+k} (k corresponds to the number of tickets that O_i has received) are no longer valid. The recipients of the notification that are not active anymore (i.e. not in the process of exchanging messages) just ignore the notification. However, the active objects send a request to $AKMS$ in order to receive new tickets. Based on experimental results (see section IV.B in [3]), DBGK outperforms its peers within a proportion of around 50% of the members in possession of the same tickets as the leaving (ejected) member. If the proportion exceeds 50%, a state of the art approach (i.e. LKH [37]) is considered to rekey the whole group. In the second case, the leaving O_i does not own any valid ticket. In this situation, forward secrecy is ensured without any rekeying operation.

4.2 Piao et al. [25] and Patsakis and Solanas [24] Schemes

Piao et al. proposed a scalable and efficient polynomial based centralized group key management protocol to secure both inter-group and intra-group communications. Nevertheless, this scheme contains security breaches. In [16], authors show that Piao et al. scheme does not ensure neither backward nor forward secrecy. In [21] authors show that Piao et al. is based on a mathematical problem computable within a reasonable amount of resources (time and computation power). An attacker can easily factorize the polynomial over a finite field and retrieve the private keys of the members, as well as the exchanged secrets.

To address these issues, Patsakis and Solanas [24] proposed a modified version of Piao et al. [25] scheme to take advantage of its efficiency while strengthening its security properties. They base their scheme on a NP-hard mathematical problem which is finding the roots of univariate polynomials modulo large composite numbers for which the factorization is not known [26]. This is in contrast with the weak mathematical problem upon which Piao et al. [25] scheme is based. Moreover, they introduce an additional virtual term in the generation of the polynomial (called salting parameter) upon every rekeying to prevent backward and forward secrecy breaches.

In DsBGK, we build upon Patsakis and Solanas [24] scheme to secure the transmission of secrets using polynomial computation instead of using encryption. Hence, efficiency and scalability are both increased. Furthermore, we enhance Patsakis and Solanas scheme to ensure forward and backward secrecy more efficiently and to increase the collusion freeness of the protocol.

5 Network Model

Our network architecture models a group of entities collaborating to achieve a defined and common goal. This group is heterogeneous, and composed of both unconstrained and constrained entities. The unconstrained entities are powerful enough to perform asymmetric primitives (e.g. desktop computers, servers, smart phones, etc.). The constrained entities are limited in terms of energy, computational, communication and storage capabilities (e.g. sensors, RFID, NFC, etc.), hence, unable to perform asymmetric primitives. Unlike in DBGK, no General Key Management Server (GKMS) is considered. Furthermore, the group is not partitioned into subgroups with Area Key Management Servers (AKMS) controlling each sub group. In fact, we consider a single logical group where the unconstrained entities play the role of controllers. These controllers maintain a consistent, distributed and open AOL (Active Object List). This list can be maintained using one of the existing solutions in the literature, such as [23]. Figure 2 illustrates our network architecture.

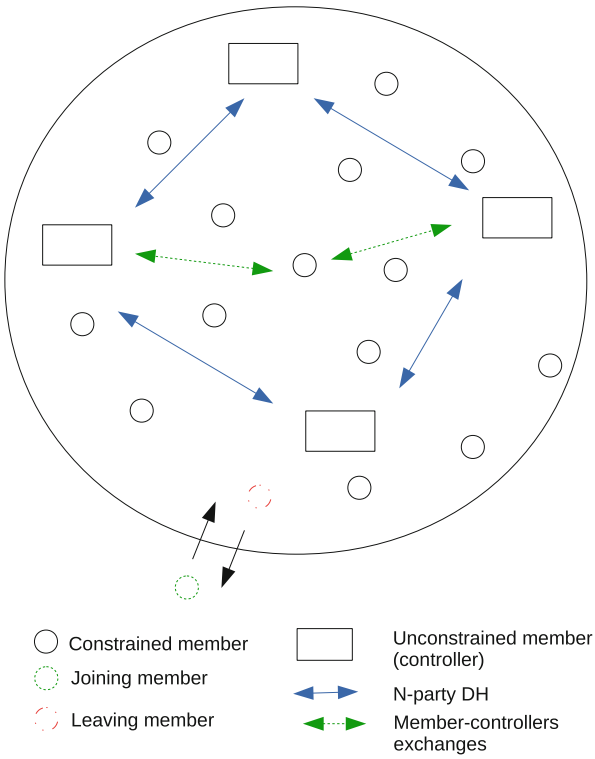

Fig. 2. Network architecture

5.1 Assumptions and Definitions

- we consider a heterogeneous group. More precisely, we assume the existence of both unconstrained members, powerful enough to perform periodic n-party Diffie-Hellman (DH) rekeyings [10], and constrained members unable to run the resource consuming n-party DH.
- the powerful entities are considered as controllers. Controllers are in charge of initiating a key update following specific events (e.g. join and leave).
- during the initialization phase, each new member is set (offline) with a private binding ID.
- during the initialization phase, at least one controller is pre-loaded (offline) with the binding ID of each new member (the ID can then be securely propagated to all controllers).
- a distributed AOL (i.e D-AOL) is maintained consistent between all controllers through the different updates.
- members are IP-enabled (6Lowpan for constrained members, and IPV6 for unconstrained members).
- we consider at a particular moment, only one active controller.

The different notations used throughout the remaining of this paper are summarized in Table 1.

6 Protocol Functioning

6.1 DsBGK General Overview

The goal of DsBGK is to establish and maintain a group key to secure communications in collaborative environments. This has to be achieved while remaining efficient and secure, ensuring both forward and backward secrecy. DsBGK is based on DBGK, we recommend the reader to refer to [3] for a comprehensive presentation of the protocol.

DsBGK proceeds within several phases. The first phase is related to the initialization of the entities. In fact, a set of unconstrained entities are designated off-line to play the role of controllers based on their capabilities. n-party DH is run within this sub-group of controllers to establish shared credentials. These latter are used to secure the communications required to update the distributed AOL (D-AOL). In addition, at least one controller is set with the secret binding ID of each new member. To become active, the new member sends a request to the active controller. The member requests one or more tickets according to its level of trust and resources capabilities. Upon successfully passing the authentication and authorization phase, the member receives the tickets along with SK (SK is only sent during the first exchange). The member will then be able to derive the group key using both the current ticket and the long term key SK. To secure the transmission of these tickets to the requesting members, the active controller builds a univariate polynomial of degree m. Upon its reception, the member computes the polynomial using its private binding ID to retrieve the

Table 1. Terminology table

Notation	Description
Group	A set of entities (members and controllers) collaborating by exchanging data in a Peer to Peer way to reach a common goal
Member (node)	An object of the group with limited resources capabilities (e.g. RFID, IP-enabled sensors, etc.)
Controller	An object of the group without hard resource constraints (e.g. personal computers, smartphones, servers, etc.)
TEK (Traffic Encryption Key)	The group key used to secure communications within the group. $TEK = F(SK, T_i)$
F	A one way function (easy to compute but hard to reverse)
SK	A long term key transmitted to each new member during its first exchange
Ticket (T_i)	Piece of data used in the generation of the TEK. T_i refers to the ticket issued for time slot i
Time slot	A defined period of time (e.g. seconds, minutes, days, etc.)
ID	Binding private identity of members. ID is used in the computation of polynomials
PublicID	Identity of the member
P(x)	Univariate polynomial modulo a composed large number n (product of two large primes $p * q$)
D-AOL	Distributed Active Object List: records all active members including the tickets they have received
SpecData	Data related to storage, processing capabilities, and trust level of members
Nslot	Number of requested time slots (tickets)

transmitted secret (i.e. tickets). The security of this scheme relies on the strength of the underlying mathematical problem. In this case, the problem comes down to finding the roots of univariate polynomials modulo large composite numbers. Upon a leave event, two situations arise. If the leaving member has not recently been active, then, no rekeying is required. However, if the leaving member is active, its tickets are no longer valid. As a result, the information stating that these tickets are no longer valid has to be propagated to the concerned members by the active controller. In the following, we present the details of DsBGK phases.

6.2 Initialization (Joining)

During this phase, the private binding ID of the member is communicated to at least one controller (typically the active controller). Upon successful authentication and authorization, the controller propagates the ID to the rest of controllers. We assume that the ID of a new members is set offline. This ID will be used to compute the received polynomials from controllers to retrieve exchanged secrets. Once the ID is set, the member is valid and can become active at any moment.

6.3 Activation

Algorithm 1 depicts the behaviour of DsBGK following a join event. After successfully joining the group, a member becomes active by requesting one (or several) tickets from the active controller. Indeed, any controller is able to deliver tickets to members, as D-AOL is distributed and maintained between all controllers. This provides a better fault tolerance compared to DBGK where only the controller, in charge of a specific area, can deliver the tickets. Upon receiving a request, a controller grants or deny the request based on several parameters related to the requesting member such as, resources capabilities and the level of trust. To secure the transmission of tickets, the active controller generates a univariate polynomial $P(x)$ modulo the product of two large prime numbers (see Algorithm 2).

$$P(x) = (x - r_1)(x - ID)(x - r_2)\dots(x - r_m) + T_i \ mod \ n$$

This polynomial represents the product of m terms plus the transmitted secret (i.e. T_i). One of the terms (i.e. $x - ID$) allows the receiving member to compute $P(ID) = 0$ to retrieve the secret. The remaining terms are set randomly.

In both Patsakis and Solanas [24] and Piao et al. [25] schemes, the terms are composed of the private credentials of the members (i.e. ID). As a result, to mitigate collusion attacks and to provide backward and forward secrecy, Patsakis and Solanas in [24] introduce the use of additional terms upon each rekeying (called salting parameters). In DsBGK, we propose to avoid using additional parameters, which can quickly increase the ratio between the polynomial degree and the actual number of users (members) within the group.

In the original Piao et al. scheme, if a new member l joins the group, this latter could breach backward secrecy (i.e. accessing data exchanged prior to the joining).

Indeed, let us consider $P_{old}(x)$ the polynomial generated before the joining, $P_{new}(x)$ the polynomial generated after the joining, n the number of users, and s the transmitted secret.

$$P_{old}(x) = (x - ID_1)\dots(x - ID_n) + s \ mod \ n$$

$$P_{new}(x) = (x - ID_1)\dots(x - ID_l)\dots(x - ID_{n+1}) + s' \ mod \ n$$

The new member m would derive the old secret s by computing:

$$s = P_{old}(x) - \frac{P_{new}(x) - s'}{x - ID_l}$$

In DsBGK, this attack would not possible, as computing $\frac{P_{new}(x)-s'}{x-ID_l}$ would give no extra knowledge considering that the terms are defined randomly (except the term that contains the ID of the recipient member) and thus vary across the different polynomials.

Furthermore, DsBGK ensures collusion freeness as the disclosure of the private ID of colluding users brings no additional knowledge to retrieve private IDs

of non-colluding members. Indeed, in each polynomial, apart from the term containing the recipient ID, the remaining terms are random and different across the polynomials. Besides, we set the degree m of the polynomial in a way to keep the factorization not easily feasible while maintaining efficiency. In [20], experimentations on MICA2 sensor showed that the computation of a polynomial of a degree up to 40 is more efficient than symmetric encryption (i.e. RC5). In DsBGK, we set m accordingly and regardless of the number of users in the group. Thus, the size of the polynomial does not grow with the growth of the number of users (members), which has a positive impact on scalability.

6.4 Leaving

To ensure forward secrecy upon a leaving event, the TEK is changed. In DsBGK, two scenarios are considered. If the leaving (ejected) member at time slot i is not in possession of valid tickets T_{i+k} (with $k \geq 0$), no rekeying is required. In fact, the leaving member will not be able to derive future TEK given the fact that group keys are partly composed of dynamic tickets. As a result, the leaving member will not have access to future communications. However, if the leaving member is in possession of tickets, the members in possession of the same tickets need to be notified. In case they are still active, they will ask for new tickets. The exchange of these secret credentials is secured using univariate polynomials generated by the active controller (see Algorithm 3).

Algorithm 1. Activation algorithm

```
1: procedure ACTIVATION (MEMBER, CONTROLLER)
2:     request ← Ticket_request{PublicID, SpecData, Nslot}
3:     Member.send(request, controller)
4:     if member is authenticated then
5:         if member is authorized then
6:             while i < number of granted tickets do
7:                 P1 ← GeneratePoly(Ti)
8:                 i ← i + 1
9:             if first activation then
10:                P2 ← GeneratePoly(SK)
11:                Controller.Send(P1, member)
12:                Controller.Send(P2, member)
13:            else
14:                Controller.Send(P1, member)
15:            Update D_AOL(controller, PublicID)
```

7 Analysis

7.1 Security Properties

Backward secrecy violation occurs when a legitimate member tries to access communications, which took place before its joining. In DsBGK, backward secrecy is

Algorithm 2. Polynomial generation algorithm

```
1: procedure GeneratePoly (secret)
2:     p ← randomly generated large prime number
3:     q ← randomly generated large prime number
4:     n ← p × q
5:     m ← fixed threshold
6:     P ← (x − ID)
7:     while i < m − 1 do
8:         r ← random_value()
9:         P ← P × (x − r) mod n
10:    P ← P + secret
11:    return(P)
```

Algorithm 3. Leaving algorithm

```
1: procedure Leaving (member, controller)
                                          ▷ retrieving tickets of the leaving member
2:     tickets ← controller.lookup(D_AOL, member)
3:     if tickets ≠ null then
4:                                        ▷ retrieving members holding the same tickets
5:         list ← controller.lookup(D_AOL, tickets);
6:         threshold ← 50% of total number of members
7:         if list.length < threshold then
8:             while list ≠ null do
9:                                              ▷ concerns only active members
10:                controller.notify(member)
11:                activation(member, controller)
12:        else                                 ▷ rekey the whole group using LKH
13:            LKH(SK)
```

ensured inherently, as joining members are not able to derive group keys which have been established prior to their joining. In fact, the group key is composed of a fixed long term key and varying tickets following each time slot. As a result, new members are unable to derive previous keys.

Forward secrecy violation occurs when a former member of the group tries to access communications, which take place after its departure from the group. In DsBGK, this property is ensured based on whether the leaving member is in possession of tickets or not. If the member is not in possession of tickets, no rekeying is required. In fact, the leaving member will not be able to derive any future group keys. However, if the member is in possession of valid tickets, using *D-AOL*, the active controller notifies only the active members which are in possession of the same tickets about their non-validity. In case the number of active members reaches a certain threshold (set experimentally to 40–50% of the total number of members in the group), the active controller relies on the state of the art LKH protocol to rekey the long term key SK. As a result, the leaving member will not be able to use its tickets to derive future group keys,

either because they are not valid anymore (and thus not used in the generation of the group key) or because the long term key has been modified.

Collusion attacks occur when two or more legitimate members collude to retrieve the security credentials of other members. In DsBGK, secret credentials are securely exchanged using univariate polynomials modulo a composite number of large primes. We ensure collusion freeness by considering variable terms, which are not based on the credentials of the users (members). Indeed, the collusion of a subset of members will not help in any form to compose polynomials with the goal of retrieving the security credentials of the remaining members. Nevertheless, this solution requires from the controller to compose a different polynomial for each member. It is worth noting, however, that the controllers are not considered as constrained members, and DsBGK main goal is to reduce the overhead with respect to the constrained members of the group.

7.2 Performance Evaluation

To analyze the performances of DsBGK and compare the results with DBGK [3], we relied on Cooja, which is the built-in network simulator of Contiki 2.7 [1]. Contiki is an open source Operating System (OS) for IP-enabled constrained devices (objects). This OS is used by the research community in several domains, such as, networked electrical systems, industrial monitoring, e-health sensors, and in Internet of Things (IoT) related applications in general. With the purpose of assessing our protocol's performances compared to DBGK's performances, we considered the same experimental setups as those used in the evaluation of DBGK. In fact, we use Tmote Sky nodes, which are equipped with the CC2420 radio chip and the MSP430 microcontroller (10k RAM, 48k Flash). Furthermore, energy consumption is computed using Powertrace tool [14]. This tool measures the time (number of ticks) during which each element (e.g. CPU, transmission, reception, etc.) of the sensor is active. This duration is combined with other data (specific to the sensor, such as the current draw, and voltage) to evaluate the energy consumption. We evaluated DsBGK performances with respect to the following metrics: storage overhead, polynomial degree, and members leave cost.

Storage overhead: In this experiment, we considered an event where a new constrained member (denoted merely by 'member' in the remaining of this analysis) joins a group. We varied the number of controllers (KMS) in order to assess the impact of additional controllers on the overhead resulting from the storage of security materials by members. The results, depicted in Fig. 3, show that for DBGK, storage overhead increases linearly with the inclusion of additional controllers. However, for DsBGK, storage overhead is steady and not related to the number of controllers. In fact, in DBGK, a pre-shared key is established between each member and each controller. This leads to a proportional dependency between the number of controllers and the number of stored keys. Indeed, in DsBGK, thanks to the use of polynomials, a pre-shared material (i.e. ID) is only set in the controller side for each additional member. Nevertheless, no material is stored in the member side. Consequently, unlike DBGK, DsBGK allows adding controllers with no impact on storage overhead.

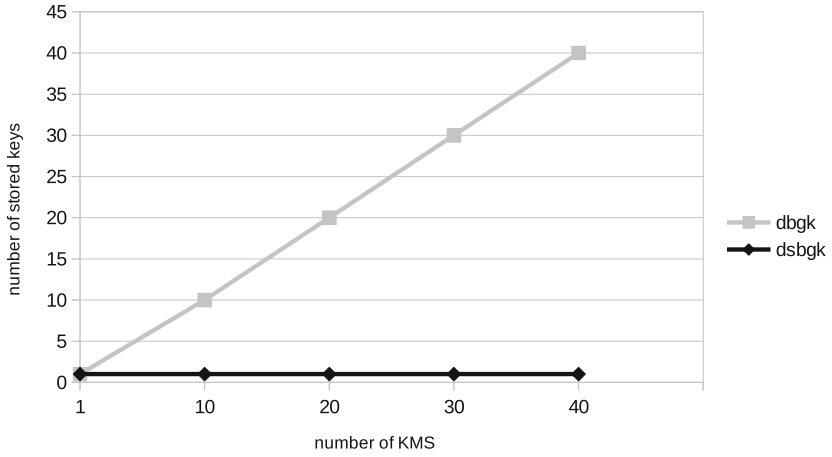

Fig. 3. Storage overhead

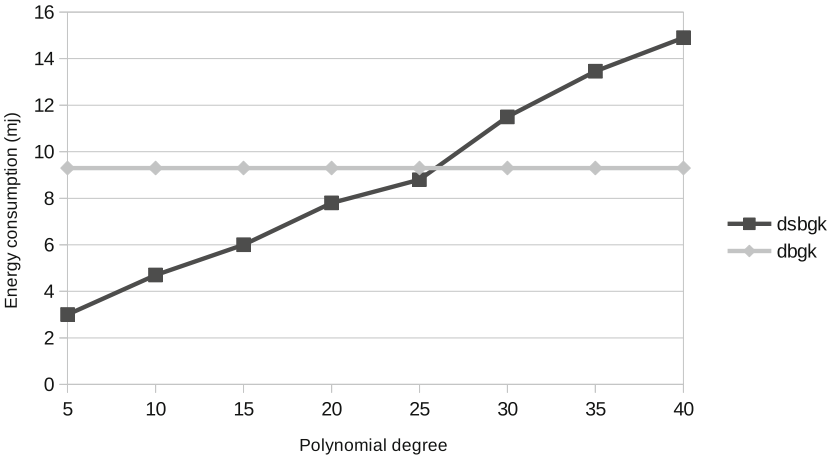

Fig. 4. Polynomial degree

The next step in our evaluation was to evaluate the impact of this gain in storage on the energy consumption induced by rekeying operations. In particular, when members leave or are ejected from the group. But first, we ran extensive simulations to set the optimal degree of the polynomial to achieve the best trade-off between security and efficiency.

Polynomial degree: We considered a group of 1000 members. We simulated a member leaving the group (or being ejected) with a proportion of 40% of remaining members holding the same tickets as the leaving member. Based on DBGK evaluation (see section IV.B in [3]), around 40–50% represents the maximum proportion above which DBGK efficiency drops and a state of the art protocol

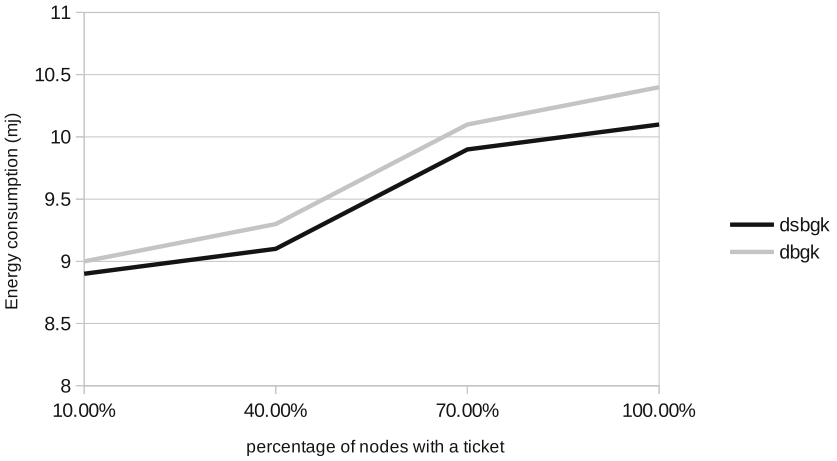

Fig. 5. Member leaving cost

(i.e. LKH [37]) is preferred to update the group key. Furthermore, $NSlot$ has been set to 20, which we consider being a realistic value. We varied the degree of the polynomial and compared energy cost with DBGK. The results presented through Fig. 4 highlight a steady raise in energy consumption with the increase of the polynomial degree. It is worth mentioning that DBGK energy cost is not impacted by polynomial degree variation, hence the constant energy consumption. Eventually, DsBGK energy cost exceeds DBGK energy cost when the degree reaches a value around 25.

Our results were slightly different compared to the experimental results presented in [20] (previously mentioned in Sect. 6.3), where performances using polynomial computation were better, up to a degree of 40. We explain this difference by the fact that we used a different sensor in our experiment (Sky mote) in addition to a different encryption primitive for DBGK (i.e. AES). Nonetheless, this variation does not alter the security foundations of DsBGK, as the NP-hard mathematical problem upon which DsBGK is based is not altered [26]. Following this experiment, we compared the energy consumptions of DBGK and DsBGK in case of a leave event to make sure that the gain in storage cost has not been achieved at the expense of other metrics.

Member leave cost: We estimated the energy cost related to the departure (or ejection) of a member in possession of a valid ticket. Similarly to DBGK's evaluation, we consider a group of users composed of 1000 members. We record several measures, while varying the proportion of members with tickets similar to those in possession of the leaving member. Moreover, we define the number of tickets requested by notified members as equal to 20 time slots (i.e. $NSlot = 20$). We depict the results in Fig. 5. It is clear that DsBGK energy consumption increases with the increase of the percentage of members in possession of the same tickets as leaving members. However, this raise in energy cost is slightly lower com-

pared to the raise noticed in DBGK energy consumption. This is mainly due to the superior efficiency of polynomial computation compared to cryptographic symmetric primitives.

Based on the obtained results, we can affirm that compared to DBGK, DsBGK provides a considerable improvement in fault tolerance and scalability. Not only this result does not incur additional overhead with respect to rekeying operations, but an improvement in energy consumption is also achieved. Back to our e-health use case scenario, presented in Sect. 2, DsBGK can be applied to efficiently secure data exchanges in such sensitive environment where the unconstrained entities (e.g. PC, smartphones, etc.) can play the role of controllers. These controllers will be in charge of efficiently managing the group key for the constrained members of the group (i.e. health related sensors). Additional controllers can be included without incurring any additional storage cost on constrained members. Thus, the failure of one or several controllers does not hinder the protocol functioning, as other controllers can take over. Furthermore, the improved efficiency is highly sought for battery powered e-health sensors. Indeed, these sensors can be planted inside human bodies. Increasing the life time of their battery would reduce the cycle of surgical interventions required for their replacement.

8 Conclusions and Perspectives

Securing distributed collaborative applications in the era of the Internet of Things relies heavily on strong and efficient group key management protocols. In this paper, we combined a polynomial based approach with our previously proposed protocol (DBGK) to propose a new protocol called DsBGK. Experimental analysis showed that DsBGK improves both fault tolerance and scalability which are highly sought in sensitive applications, such as e-health systems. Energy gains are also achieved, which makes DsBGK suitable for heterogeneous, and dynamic collaborative groups. We plan to further investigate DsBGK security strength by thoroughly assessing properties such as data integrity, data authentication, and data confidentiality through an implementation using automated formal validation tools (e.g. Avispa, Scyther). In addition, we are currently investigating a lightweight blockchain based scheme to allow sensors authenticating genuine controllers.

References

1. The Contiki Operating System. http://www.contiki-os.org
2. Abdmeziem, M.R., Tandjaoui, D.: An end-to-end secure key management protocol for e-health applications. Comput. Electr. Eng. **44**, 184–197 (2015)
3. Abdmeziem, M.R., Tandjaoui, D., Romdhani, I.: A decentralized batch-based group key management protocol for mobile internet of things (DBGK). In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), pp. 1109–1117. IEEE (2015)

4. Abdmeziem, M.R., Tandjaoui, D., Romdhani, I.: Architecting the internet of things: state of the art. In: Koubaa, A., Shakshuki, E. (eds.) Robots and Sensor Clouds. SSDC, vol. 36, pp. 55–75. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-22168-7_3
5. Abdmeziem, M.R., Tandjaoui, D., Romdhani, I.: A new distributed MIKEY mode to secure e-health applications. In: Proceedings of the International Conference on Internet of Things and Big Data, IoTBD, vol. 1, pp. 88–95. SciTePress (2016)
6. Abdmeziem, M.R., Tandjaoui, D., Romdhani, I.: Lightweighted and energy-aware MIKEY-ticket for e-health applications in the context of internet of things. Int. J. Sens. Netw. (2017, in press)
7. Balenson, D., McGrew, D., Sherman, A.: Key management for large dynamic groups: one-way function trees and amortized initialization. Internet-Draft, February 1999
8. Ballardie, A.: Scalable multicast key distribution. RFC 1949, May 1996
9. Briscoe, B.: MARKS: zero side effect multicast key management using arbitrarily revealed key sequences. In: Rizzo, L., Fdida, S. (eds.) NGC 1999. LNCS, vol. 1736, pp. 301–320. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-540-46703-8_19
10. Challal, Y., Seba, H.: Group key management protocols: a novel taxonomy. Int. J. Inf. Technol. **2**(1), 105–118 (2005)
11. Daghighi, B., Kiah, M., Shamshirband, S., Rehman, M.: Toward secure group communication in wireless mobile environments: issues, solutions, and challenges. J. Netw. Comput. Appl. **50**, 1–14 (2015)
12. Di Pietro, R., Mancini, L.V., Jajodia, S.: Providing secrecy in key management protocols for large wireless sensors networks. Ad Hoc Netw. **1**(4), 455–468 (2003)
13. Dini, G., Lopriore, L.: Key propagation in wireless sensor networks. Comput. Electr. Eng. **41**, 426–433 (2015)
14. Dunkels, A., Eriksson, J., Finne, N., Tsiftes, N.: Powertrace: network-level power profiling for low-power wireless networks (2011)
15. Harney, H., Muckenhirn, C.: Group key management protocol (GKMP) architecture. RFC 2093, July 1997
16. Kamal, A.A.: Cryptanalysis of a polynomial-based key management scheme for secure group communication. IJ Netw. Secur. **15**(1), 68–70 (2013)
17. Keoh, S.L., Kumar, S.S., Tschofenig, H.: Securing the internet of things: a standardization perspective. IEEE Internet Things J. **1**(3), 265–275 (2014)
18. Kim, Y., Perrig, A., Tsudik, G.: Tree-based group key agreement. ACM Trans. Inf. Syst. Secur. (TISSEC) **7**(1), 60–96 (2004)
19. Lee, P., Lui, J., Yau, D.: Distributed collaborative key agreement and authentication protocols for dynamic peer groups. IEEE/ACM Trans. Netw. **14**(2), 263–276 (2006)
20. Liu, D., Ning, P.: Security for Wireless Sensor Networks, vol. 28. Springer Science & Business Media, Heidelberg (2007). https://doi.org/10.1007/978-0-387-46781-8
21. Liu, N., Tang, S., Xu, L.: Attacks and comments on several recently proposed key management schemes. IACR Cryptology ePrint Archive 2013:100 (2013)
22. Mittra, S.: Iolus: a framework for scalable secure multicasting. ACM SIGCOMM Comput. Commun. Rev. **27**(4), 277–288 (1997)
23. Oster, G., Urso, P., Molli, P., Imine, A.: Data consistency for P2P collaborative editing. In: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, pp. 259–268. ACM (2006)
24. Patsakis, C., Solanas, A.: An efficient scheme for centralized group key management in collaborative environments. IACR Cryptology ePrint Archive 2013:489 (2013)

25. Piao, Y., Kim, J., Tariq, U., Hong, M.: Polynomial-based key management for secure intra-group and inter-group communication. Comput. Math. Appl. **65**(9), 1300–1309 (2013)
26. Plaisted, D.A.: New NP-hard and NP-complete polynomial and integer divisibility problems. Theor. Comput. Sci. **31**(1–2), 125–138 (1984)
27. Rafaeli, S., Hutchison, D.: Hydra: a decentralized group key management. In: 11th IEEE International WETICE: Enterprise Security Workshop, June 2002
28. Rafaeli, S., Hutchison, D.: A survey of key management for secure group communication. ACM Comput. Surv. (CSUR) **35**(3), 309–329 (2003)
29. Setia, S., Koussih, S., Jajodia, S., Harder, E.: Kronos: a scalable group re-keying approach for secure multicast. In: Proceedings IEEE Symposium on Security and Privacy, pp. 215–228 (2000)
30. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
31. Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A.: Security, privacy and trust in internet of things: the road ahead. Comput. Netw. **76**, 146–164 (2015)
32. Sicari, S., Rizzardi, A., Miorandi, D., Coen-Porisini, A.: Internet of things: security in the keys. In: Proceedings of the 12th ACM Symposium on QoS and Security for Wireless and Mobile Networks, pp. 129–133. ACM (2016)
33. Tang, P.C., Ash, J.S., Bates, D.W., Overhage, J.M., Sands, D.Z.: Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption. J. Am. Med. Inform. Assoc. **13**(2), 121–126 (2006)
34. Veltri, L., Cirani, S., Busanelli, S., Ferrari, G.: A novel batch-based group key management protocol applied to the internet of things. Ad Hoc Netw. **11**(8), 2724–2737 (2013)
35. Wang, W., Bhargava, B.: Key distribution and update for secure inter-group multicast communication. In: Proceedings of the 3rd ACM Workshop on Security of ad Hoc and Sensor Networks, pp. 43–52. ACM (2005)
36. Wang, W., Wang, Y.: Secure group-based information sharing in mobile ad hoc networks. In: IEEE International Conference on Communications, ICC 2008, pp. 1695–1699. IEEE (2008)
37. Wong, C., Gouda, M., Lam, S.: Secure group communications using key graphs. IEEE/ACM Trans. Netw. **8**(1), 16–30 (2000)