Securing IoT-based Groups: Efficient, Scalable and Fault-tolerant Key Management Protocol

Mohammed Riyadh Abdmeziem, François Charoy

Securing IoT-based Groups: Efficient, Scalable and Fault-tolerant Key Management Protocol

MOHAMMED RIYADH ABDMEZIEM[1]*, FRANCOIS CHAROY[1]

Université de Lorraine (Inria-CNRS-LORIA), Nancy, France

Group key management protocols are crucial in establishing secured communication channels for collaborative IoT-based groups. The Internet of Things (IoT) dimension includes additional challenges. In fact, resource constrained members within dynamic and heterogeneous groups are unable to run existing group key protocols. Furthermore, these protocols need to be scalable and fault tolerant to suit growing and sensitive groups. To face these issues, we enhance our previously proposed protocol called Decentralized Batch-based Group Key protocol (DBGK). Using polynomial computation to secure data exchanges, we considerably improve its scalability, fault tolerance and collusion freeness properties. This gain is achieved thanks to the ability to include additional unconstrained members (controllers) while inducing a very limited cost on the constrained members. Furthermore, we include an energy preserving blockchain-based mechanism to authenticate group members credentials in a distributed manner. To assess our new protocol called DiStributed Batch-based Group Key protocol (DsBGK), we performed a detailed theoretical security analysis to evaluate its behaviour against well studied attacks in the literature. Furthermore, we validated this analysis using a formal validation tool. To evaluate DsBGK performances, we performed extensive simulations. We proceeded by comparing DsBGK in term of energy cost, first, with DBGK, then with other analogous protocols from the literature. The results confirmed the security soundness of DsBGK, in addition to an improved energy efficiency compared to its peers.

Key words: Collaborative groups, Internet of Things (IoT), Security,

* email: mohammed-riyadh.abdmeziem@loria.fr

Group key management, Polynomial computation, Blockchains, Contiki, Avispa.

1 INTRODUCTION

Collaborative applications entered a new era with the advent of the Internet of Things (IoT), and its use in information systems. Various objects from our environment are enhanced with perception and actuation capabilities. These pervasive devices are autonomous enough to act independently relying on the perceived environment targeting a specific goal. Indeed, building on data collected from both users and objects resulted in numerous applications ranging across several domains such as healthcare, transportation and military environments [8]. In these sensitive applications, strong guarantees in terms of data confidentiality, and users privacy need to be ensured. The distributed nature of such pervasive 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 [55] [10] [6].

In order to highlight these security challenges through a relevant use case scenario, let us consider a personal health record system (PHR) [56] (FIGURE. 1) (also called e-health application). Indeed, this scenario is a typical illustration of a collaboration on sensitive and private data among health-care personal, insurers, caregivers, patients and sensors. The goal being to maintain the patient's medical status, health history and treatment. To access and modify shared data, some participants (e.g. medical staff) collaborate using unconstrained devices, such as Personal Computers (PC) and smartphones, while sensors planted in or around the human, collaborate by communicating their sensed data to other users or directly editing patient's medical record. It is worth highlighting that the different involved sensors are considered as constrained since they have limited computing power and may operate on battery. As a result, a gap is created regarding the ability of all entities to run the required security protocols. Furthermore, medical staff can also control the sensors (trigger or stop the sensing of a particular physiological data), and include additional sensors to the collaboration. Hence, members can join and leave the collaboration around the medical record as the situation of the patient evolves. In this context, there is an obvious need to provide decentralized, secure, energy-aware, privacy preserving and scalable group key management protocols to secure communications among people and sensors (objects).

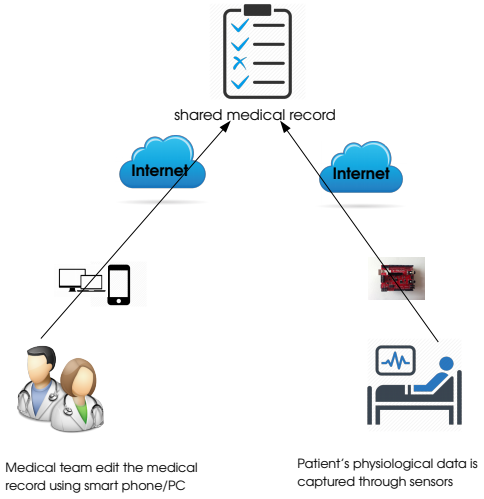

FIGURE 1
E-health use case scenario

Group key management is challenging in this environment. In fact, collaborative groups encompass heterogeneous members with different requirements and resources capabilities [31]. This gap can disrupt end-to-end communications. As a matter of fact, constrained members with limited processing power and storage space can not run heavy cryptographic primitives [9]. 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) [54].

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 proto-

col 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) [7]. 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. Keying materials are distributed to joining members based on their resources capabilities. Experiments showed that DBGK [7] is energy efficient and outperforms similar existing protocols in the literature.

Although efficient and secure, DBGK relies on unconstrained key management servers (called controllers) to maintain the group key. Each member of the group share credentials with each controller. As a result, including additional controllers to improve fault tolerance would impose a high storage overhead on constrained members. Furthermore, a manual intervention is required on the constrained members to store the shared security credentials. This might not be convenient in some cases like in e-health applications, where sensors might be planted inside human bodies. Consequently, DBGK is not appropriate 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 basic functioning of DBGK, while significantly improving both fault tolerance and scalability. In fact, the contributions of DsBGK can be summarized as follows:

- Instead of using encryption to secure communications between constrained members and controllers, we propose a polynomial based scheme inspired from [45] and [44].

- This scheme allows including additional controllers with very limited impact on constrained members regarding storage cost.

- Furthermore, no manual intervention is required on the constrained node side.

- Taking into account the highly scalable nature of IoT applications, we enhance the polynomial based scheme proposed in [45] and [44] to be

able to handle a high number of members while keeping a constant polynomial degree.

- Taking into account the sensitivity of IoT applications, we further enhance our polynomial scheme to improve collusions freeness. In fact, the disclosure of private credentials from colluding users brings no additional knowledge to retrieve private credentials of non colluding members.

- Taking into account the distributive nature of IoT, we complete DsBGK with a lightweight blockchain-based solution to allow constrained members authenticate the controllers. This would prevent malicious controllers from tampering the identity of constrained members, which could lead to data leakage.

To assess the security properties of DsBGK, firstly, we theoretically analyzed its behavior when faced with common security threats, secondly, we validated this analysis by implementing a specification of DsBGK within a formal validation tool called *Avispa* [1]. In fact, we considered several backend techniques to assess DsBGK security properties. In addition, we analyzed the obtained results using a graphical tool called SPAN [27]. This analysis confirmed the soundness of DsBGK in terms of data confidentiality, data authentication, and data integrity. To compare the efficiency of DsBGK with other protocols in the literature, we conducted extensive experiments using the built-in network simulator of the Contiki OS, namely, Cooja [2]. In a first step, we compared DsBGK with DBGK. 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. In a second step, we compared DsBGK with analogous protocols in the literature, namely, Veltri et al [58], MARKS [15], and LKH [61]. The results showed that DsBGK outperforms its peers in terms of energy efficiency following various membership events such as, members departure, and members joining.

The remaining of the paper is organized as follows. In section 2, we discuss in detail, existing solutions in the literature, as well as, the required background. In section 3, first, we present our network model, along with our assumptions and the used notations. Then, we thoroughly present the functioning of DsBGK, before assessing its security properties in section 4.

In section 5, we introduce and analyze the simulation results. In section 6, we provide some final remarks.

2 RELATED WORK

In this section, we review the main categories under which group key management protocols are usually categorized [21] [48], 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 [29] 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 [58] 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 [61], later improved by the One-way Function Tree protocol [13] 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 [53]. More recently, authors in [59] [60] 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 [45] 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 [21]: *common Traffic Encryption Key (TEK) per area* [15] [47], and *independent TEK per area* [45] [41]. 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 [16] by including *time-driven* rekeying subcategory [15] [52] and *membership-driven* rekeying subcategory [47] [14]. 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 [32] [34]. 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 [23] 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 [22] 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.

DBGK [7]

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 [7]), DBGK outperforms its peers. This is true when the proportion of members in possession of the same tickets as the leaving (ejected) member does not exceed 50%. If the proportion exceeds 50%, a state of the art approach (i.e. LKH [61]) 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.

Piao et al [45] and Patsakis et al [44] 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 [30], authors show that Piao et al scheme does not ensure neither backward nor forward secrecy. In [37] 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 et al [44] proposed a modified version of Piao et al [45] 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 [46] . This is in contrast with the weak mathematical problem upon which Piao et al [45] 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 et al [44] scheme to secure the transmission of secrets using polynomial computation instead of using encryption. Furthermore, we enhance Patsakis et al scheme to ensure forward and backward secrecy more efficiently and to increase the collusion freeness of the protocol.

3 PROTOCOL FUNCTIONING

3.1 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

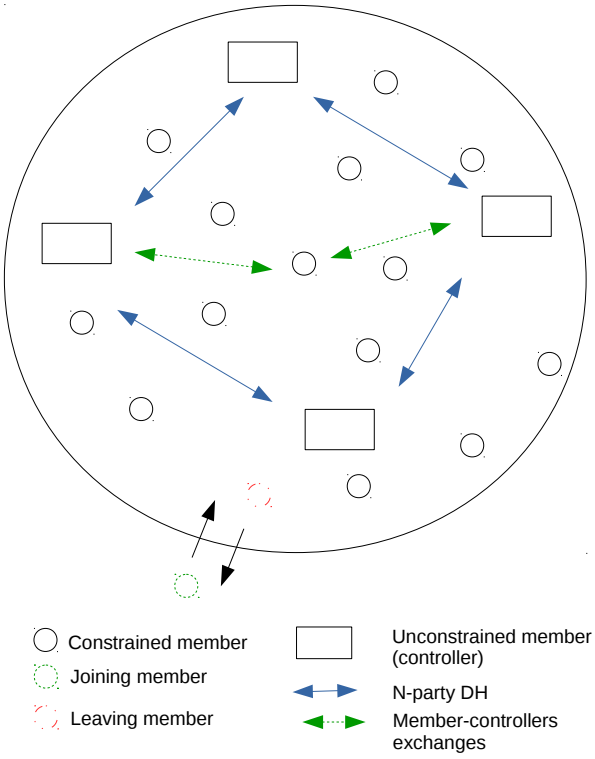

FIGURE 2
Network model

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)

TABLE 1
Terminology table

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 convergent using one of the existing solutions in the literature, such as [43]. FIGURE. 2 illustrates our network model.

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 [16], 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, at least one controller is pre-loaded (offline) with the binding ID of each new member. The binding is appended to a blockchain data structure and propagated to the remaining controllers (more details in section 3.4).

- 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.

3.2 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 [7] for a comprehensive presentation of the protocol.

DsBGK proceeds within several phases (see FIGURE 3). 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

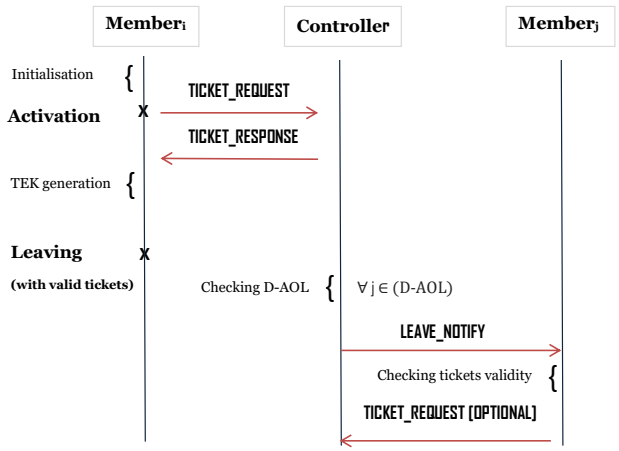

FIGURE 3
DsBGK signaling flow

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. Moreover, we propose a system based on a private blockchain to record and authenticate the binding between the private and the public ID of the constrained members (see section 3.3 and section 3.4).

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 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.

3.3 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. As a result, it is important to notice, that the inclusion of additional controllers implies no operation on the constrained members, which increases **fault tolerance**.

It is worth highlighting, that in case the binding between the public ID and the private ID (PublicID, ID) is maliciously altered, the group key secrecy would be in jeopardy. Indeed, secret credentials might be transmitted by the

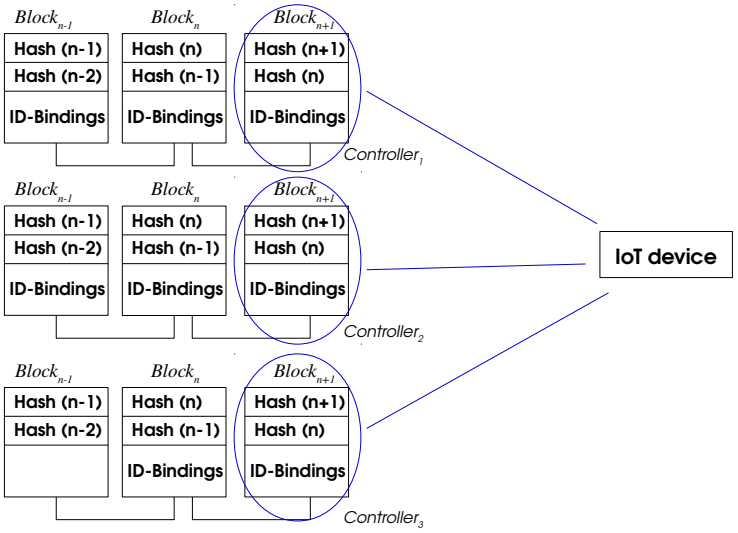

FIGURE 4
A lightweight blockchain mechanism to authenticate ID bindings

controllers to illegitimate members. In the following subsection, we present a blockchain-based mechanism to ensure the validity of ID bindings.

3.4 ID bindings: a blockchain-based solution

A blockchain is a fully distributed list of blocks. An analogy is often made with a distributed ledger that is replicated within all members of the network. Each block of this blockchain records a set of transactions.

Among other information, each block contains a hash of its own transactions in the form of a Merkel tree [40]. Moreover, the cryptographic hash of the previous block is also included. As a result, as soon as a transaction is maliciously altered within any block, it can easily be detected, as the hashes are propagated and would inevitably diverge. Furthermore, before appending a new block to the blockchain, a special type of members (called miners)

perform a computational intensive task called proof-of-work to validate this new block. Other validation techniques can be used following the blockchain implemntation (e.g. proof-of-stake, proof-of-burn). Indeed, several implementations of blockchains currently exist, particularly in the domain of cryptocurrencies such as Bitcoin and Ethereum. Besides, a blockchain can either be public or private. A public blockchain does not enforce any access control mechanism for the members of the network and all transactions are thus publicly available. However, a private blockchain only encompasses authorized members [11] [12].

The distributed, immutability, and trustless nature of blockchain mechanisms can be used to ensure the correctness of the ID binding in our protocol. In fact, we propose to consider a private blockchain replicated within all controllers. These latter are not constrained and can withstand the computing and storage overhead induced by blockchain related operations. The transactions of the blocks represent the ID bindings. Doing so, if a controller is compromised and tries to alter the ID bindings, it can easily be detected by the rest of controllers. Consequently, the corrupted controller would no longer receive the latest blocks of the blockchain. Furthermore, controllers can be authenticated by sending the latest block of the blockchain to the constrained members of the group (i.e. IP-enabled sensors). The constrained members would then check the hash of the latest block which must match the hashes received from other controllers (see FIGURE 4).

3.5 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 ($TICKET\_REQUEST$ message). Indeed, any controller is able to deliver tickets to members ($TICKET\_RESPONSE$ message), 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)

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

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 unlike both Patsakis et al [44] and Piao et al [45] schemes. Indeed, in these latter, the terms are composed of the private credentials of the members (i.e. ID). To mitigate collusion attacks and to provide backward and forward secrecy, additional terms are included upon each rekeying (called salting parameters), which can rapidly increase the ratio between the polynomial degree and the actual number of users (members) within the group. As a result, it is important to mention that in DsBGK the size of the polynomial does not grow with the growth of the number of users (members), which has a positive impact on **scalability**.

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.

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

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

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 [36], 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).

3.6 Leaving

To ensure forward secrecy upon a leaving event, the TEK is changed. In Ds-BGK, two scenarios are considered. If the leaving (ejected) member at time slot i is not in possession of valid tickets T_{i+k} (with $k >= 0$) , 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 ($LEAVE\_NOTIFY$ message). 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: $request \leftarrow Ticket\_request\{PublicID, SpecData, Nslot\}$
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: $P_1 \leftarrow GeneratePoly(T_i)$
8: $i \leftarrow i + 1$
9: **if** $first\ activation$ **then**
10: $P2 \leftarrow GeneratePoly(SK)$
11: $Controller.Send(P1, member)$
12: $Controller.Send(P2, member)$
13: **else**
14: $Controller.Send(P1, member)$
endif
15: $Update\ D\_AOL(controller,\ PublicID)$
endif
16: **endif**

Algorithm 2 Polynomial generation algorithm

procedure GENERATEPOLY (SECRET)
  $p \leftarrow randomly\ generated\ large\ prime\ number$
  $q \leftarrow randomly\ generated\ large\ prime\ number$
  $n \leftarrow p \times q$
  $m \leftarrow fixed\ threshold$
  $P \leftarrow (x - ID)$
  while $i < m - 1$ **do**
    $r \leftarrow random\_value()$
    $P \leftarrow P \times (x - r)\ mod\ n$
  $P \leftarrow P + secret$
  $return(P)$

Algorithm 3 Leaving algorithm

procedure LEAVING (MEMBER, CONTROLLER)
  ▷ retrieving tickets of the leaving member
  $tickets \leftarrow controller.lookup(D\_AOL, member)$
  if $tickets \neq null$ **then**
    ▷ retrieving members holding the same tickets
    $list \leftarrow controller.lookup(D\_AOL, tickets)$;
    $threshold \leftarrow 50\%\ of\ total\ number\ of\ members$
    if $list.length < threshold$ **then**
      while $list \neq null$ **do**
        ▷ concerns only active members
        $controller.notify(member)$
        $activation(member, controller)$
      else ▷ rekey the whole group using LKH
        $LKH(SK)$
      endif
  endif

4 SECURITY ASSESSMENT

4.1 Security properties

To study DsBGK security properties, let us consider the personal health record (PHR) [56] (FIGURE. 1) use case application presented in the introduction. Indeed, this latter is vulnerable to a multitude of security threats [5] [35]. In the following, we discuss how DsBGK behaves when faced with common attacks which can threaten the safety of PHR systems.

Health related applications are particularly sensitive, due to the very private nature of the exchanged data. Confidentiality is a crucial property to be preserved for the sake of users privacy. Any data breach would seriously jeopardize the adoption of these applications. In DsBGK, confidentiality is ensured through the generation of a univariate polynomial. The hard mathematical problem to be solved in this case is retrieving the roots of univariate polynomials modulo large composite numbers [44]. It is important to highlight that the factorization of these numbers is not known. Indeed, the composite numbers can be set relying on the same properties as those used in other security protocols such as RSA [50]. Doing so, the factorization of these numbers would be nearly impossible (not feasible within a reasonable timeframe). As a result, this mathematical problem cannot be transformed to a trivial problem (i.e. finding the roots of univariate polynomials modulo prime numbers), which can be solved in a reasonable time.

Backward secrecy violation occurs when a legitimate member tries to access health related data, that has been exchanged before its joining. In DsBGK, backward secrecy is 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. Hence, new members are unable to derive previous keys.

Forward secrecy violation occurs when a former member of the group tries to access communications, which took place after its departure from the group. In DsBGK, the protection against this violation is 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, 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.

PHR systems can dramatically be impacted by replay attacks. In fact, delayed or outdated data can trigger inappropriate medical responses which might end up with unfortunate consequences. DsBGK provides key freshness by including nonces in each exchanged message. These nonces can be considered as random numbers, sequence numbers, or timestamps according to members capabilities. Using random numbers, members store all previously received numbers. Doing so, following the reception of a message, the member verifies if the received random number has already been received previously. This solution requires the storage of all received numbers, which can lead to a considerable storage overhead, especially for highly constrained members. Sequence numbers, on the other hand, can spare members from storage constraints. In fact, each message would contain an incremental number, so as recent messages would present a greater number compared to older ones. Nevertheless, in case a member fails, there would be no way of keeping track of the current sequence. Besides, timestamps can also be considered in spite of their unsuitability with constrained members as a high amount of resources are required to ensure time synchronization.

It is obvious that using clock synchronization between constrained members and controllers is not adapted. Nevertheless, timestamps can be envisioned as a countermeasure to be used between unconstrained controllers. Furthermore, random numbers and sequence numbers are more suited to protect the messages involving constrained members. Providing a strong reliability (remote probability of failures), sequence numbers are the more appropriate choice, especially, for members with limited storage capabilities. In case

storage capabilities are not a concern, random numbers can be envisioned. In a nutshell, protecting DsBGK from replay attacks is ensured by combining the discussed solutions based on the specificities of the network model.

In Sybil attacks [25] [39], legitimate members can claim several fake identities. This can engender disastrous consequences for health related systems (e.g. PHR). In fact, a malicious user can send altered data using feigned identities. Consequently, either real emergency situations are missed, or false emergency alarms are triggered. DsBGK is protected against Sybil attacks. In fact, there is no possibility for a malicious member to perform a Sybil attack unless the controller (assumed to be a trusted entity) is corrupted. Indeed, the controller secures its exchanges with members based on their private ID, which is binded with their publicID. Furthermore, once the group key is established, all exchanged messages are authenticated and contain the identity of the sender. In addition, before any further processing, the controller checks its access control policy with respect to any new joining member.

In PHR systems, the availability of health data must be resilient against Denial of Service (DoS) attacks. In DsBGK, the different messages are authenticated through the binding public ID-private ID before data processing is performed. Indeed, local states are not established prior to authentication. In addition, traditional solutions can also be used. For instance, rate-limiting techniques which abort the execution of the protocol following the detection of corrupted messages can also be envisioned. Besides, attacks that aim to drain the energy power of the members constitute another point of interest regarding the PHR systems threat model. The De-synchronization attack alters the sequence number of the different messages. Doing so, messages can be retransmitted indefinitely. Clearly, this would lead to the draining of energy resources. The main countermeasure against this kind of attacks is to ensure message integrity. In DsBGK, once the group key is set, we advocate to protect all messages by appending Message Authentication Codes (MAC).

It is worth mentioning that PHR systems can also be targeted by routing attacks. DsBGK does not protect the group from these kind of attacks. Its primary goal is to manage a master group key, which is then used to secure data transmission. Intrusion Detection Systems (IDS) [49] [33] should be considered to address routing related threats.

4.2 Formal validation

Many solutions have been used to formally validate security protocols. Among others, model checking [20] is usually considered to evaluate finite-state-concurrent protocols (e.g. communication protocols). In general, verification

tools are used to thouroughly look for all eventual execution paths that satisfy some desired properties stated in a protocol specification. In the literature, the validation of several security protocols has been carried out using model checking [57] [28]. In addition, model checking has also been at the basis of the development of a multitude of validation tools [1] [4] [3]. In the following, we discuss the advantages of considering model checking techniques instead of traditional techniques built around deductions, testing, and simulations:

- Enables rapid and automated assessment using various model checking tools. Thus, users are spared from prototyping their protocol.

- Enables users to assess each single stage of the execution process. As a result, users have the possibility to spot any failure accurately. Nevertheless, simulation techniques only provide a general overview of the protocol functioning. Moreover, parts of the flaws can stay hidden, and only discovered at production stage.

AVISPA (Automated Validation of Internet Security Protocol and Applications)[1] is a tool used to assess and validate security protocols. Avispa is based on a set of model checkers running in the back-end, with which interaction is made through a shared front-end. The Dolev-Yao intruder model [24] is considered to model messages interception and altered data insertion. The model engages analytical rules to evaluate the security soundness of the protocol. If a flaw is detected, the tool produces a log detailing the different stages of the attack. In the state of the art, the security assessment of a multitude of protocols has been carried out using Avispa [19] [38] [17] [51]. Furthermore, several standard based protocols such as IKE, TLS, and AAA maintained by the Internet Engineering Task Force (IETF) have been analyzed using Avispa. In fact, flaws have been spotted in some of them [42] [1].

We performed a formal validation of DsBGK based on Avispa to assess its security properties such as authentication, delivery proof and replay protection. It is worth noting here that since DsBGK confidentiality is based on polynomial computation instead of traditional symmetric encryption, this property has not been considered in the Avispa evaluation. Indeed, the strength and resistivity of the polynomial is directly related to the factorization problem (see section 4.1). In general, Avispa (and other formal validation tools for that matter) does not assess the strength of the cryptographic primitives but rather focus on the messaging flow.

In Avispa, protocol's behaviors are specified using a role-based language (i.e. High Level Protocol Specification Language (HLPSL)) [18]. A *basic*

Notation	Member
A	Member
B	Controller
ID_X	Identity of x
$Nonce_X$	Nonce generated by x
$Data_X$	data related to storage, processing capabilities, and trust level of x
$Grant - Notify_X$	access control decision regarding x
$Ticket - Request_X$	tickets request by x
$Ticket - Response_X$	tickets response for x
Move-Request	request of leave
Leave-Notify	acknowledgement of the leave request, or notification of an ejection
$Message_{SEC}$	message confidentiality is secured through polynomial computation

TABLE 2
Alice-Bob terminology for HLPSL specification

role module contains the actions of the various entities. The interactions between them are specified through the composition of a multitude of *basic roles*, hence forming a *composed role*. Besides, the security properties to be assessed are mentioned in the *goal section*. It is worth mentioning that Avispa uses several automated analysis techniques: on-the-fly model-checker (OFMC), constraint-logic based attack searcher (CL-AtSe), and SAT-based model checker (SATMC) are currently in use.

Firstly, we specified the actions of the various entities in a *basic role*. Secondly, we described the participants interactions in a *composed role*. In our validation, we covered message exchanges related to joining events that trigger a ticket request. We also covered leave/ejection events which trigger a new ticket request. For the sake of simplicity, in this section, our modeling is introduced through Alice-Bob ($A - B$) notation instead of raw HLPSL specification (TABLE 1 provides more details about the notations).

The mapping with DsBGK's concepts, along with the used notations are defined in TABLE 2 :

- *A: Member (i.e. object)*

- *B: Controller*

Joining exchanges:

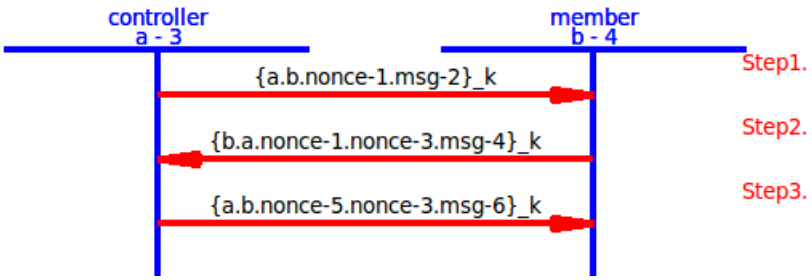

FIGURE 5
DsBGK leave process using SPAN animation tool

The message exchanges have been modeled as follows:

* $A->B:\{ID_A, Data_A, Nonce_A\}$
* $B->A:\{ID_B, ID_A, Grant-Notify_A, Nonce_A, Nonce_B\}_{SEC}$
* $A->B:\{ID_A, ID_B, Ticket-Request_A, Nonce_A, Nonce_B\}_{SEC}$
* $B->A:\{ID_A, ID_S, Ticket{-}Response_A, Nonce_A, Nonce_B\}_{SEC}$

Leaving exchanges:

The message exchanges have been modeled as follows:

* [Optional] $A->B:\{ID_A, ID_B, Move-Request, Nonce_A\}_{SEC}$
* $B->A:\{ID_B, ID_A, Leave-Notify, Nonce_B\}_{SEC}$
* $A->B:\{ID_A, ID_B, Ticket-Request_A, Nonce_A, Nonce_B\}_{SEC}$
* $B->A:\{ID_B, ID_A, Ticket{-}Response_A, Nonce_B, Nonce_A\}_{SEC}$

Once the modeling completed, we inserted the specification as input to Avispa, and checked DsBGK correctness relying on a protocol animation tool (i.e. SPAN) [27]. This tool has been proposed to offer a graphical support for the writing and analysis of Avispa specifications (See FIGURE 5). We launched several Avispa backends like OFMC, $CL-AtSe$, SATMC and

```
user@instant-contiki:~/avispa-1.1$ avispa DsBGK_Joining2.hlpsl --ofmc
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  ../avispa-1.1/testsuite/results/DsBGK_Joining2.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 123.66s
  visitedNodes: 10035 nodes
  depth: 17 plies
```

FIGURE 6
Joining: Avispa output (OFMC)

```
user@instant-contiki:~/avispa-1.1$ avispa DsBGK_Leaving.hlpsl --ofmc
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  ../avispa-1.1/testsuite/results/DsBGK_Leaving.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 7.17s
  visitedNodes: 902 nodes
  depth: 12 plies
```

FIGURE 7
Leaving: Avispa output (OFMC)

TA4SP to assess our security goals. Furthermore, we considered the default Dolev-Yao intruder model. This model simulates an attacker which is in full control of the network. Both sent and received messages can be intercepted, analyzed, modified, or sent to malicious entities.

With respect to each back-end, Avispa produces a report that depicts the results of the simulation. In our case, the results highlighted that DsBGK is "SAFE" against OFMC (for illustration purposes, see FIGURE 6 for the joining exchanges, and FIGURE 7 for the leaving exchanges), and $CL - AtSe$. Nevertheless, the results indicated by the TA4SP model were "INCONCLUSIVE". Based on Avispa documentation [1], an inconclusive result does not indicate a flaw. Relying on both our analysis and on Avispa output, we can affirm that DsBGK guarantees the main security properties namely, confidentiality, integrity and authentication.

Following the security assessment, we performed an evaluation of DsBGK energy cost. In the next section, we discuss the obtained results.

5 PERFORMANCE EVALUATION

To evaluate DsBGK energy cost, we went through two phases. In the first phase, we compared DsBGK with DBGK [7], while in the second phase, we proposed a broader comparison by including relevant state of the art schemes (MARKS[15], LKH [61], and Veltri et al [58]). In our evaluation, we considered both storage and energy costs upon common events in collaborative IoT groups (i.e. joining,sending/receiving messaging, and leaving). We performed this evaluation using Cooja, which is the built-in network simulator of Contiki 2.7 [2]. Contiki is an open source Operating System (OS) for IP-enabled constrained devices (objects). It is widely used in the research community for Internet of Things (IoT) related applications. Examples are networked electrical systems, industrial monitoring, and e-health systems. In our experimental setup, we considered 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 [26]. 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.

5.1 First phase (DsBGK vs DBGK)

In this phase, we evaluated DsBGK performances against DBGK with respect to the following metrics: storage overhead, polynomial degree, and members leave.

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 FIGURE. 8, show that for DBGK, storage overhead increases linearly with the inclusion of additional controllers. However, for DsBGK, storage overhead is steady and is 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. It is clear that the ability of including additional controllers has a strong positive impact on scalability and fault tolerance.

The next step in our evaluation was to evaluate the impact of this gain in storage cost on the energy consumption induced by rekeying operations. In particular, when members leave (or are ejected from the group). To do so, we need to set our polynomial degree to achieve the best trade-off between security and efficiency. Hence, before carrying on with our evaluation, in the following we detail our experiments to set the optimal degree.

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 [7]), around 40-50 $\%$ represents the maximum proportion above which DBGK efficiency drops and a state of the art protocol (i.e. LKH[61]) 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 FIGURE. 9 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

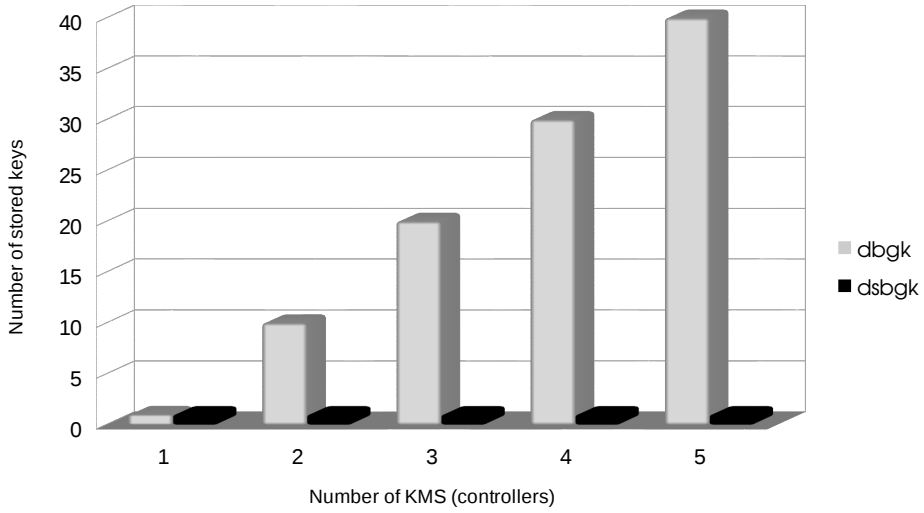

FIGURE 8
Storage cost comparison

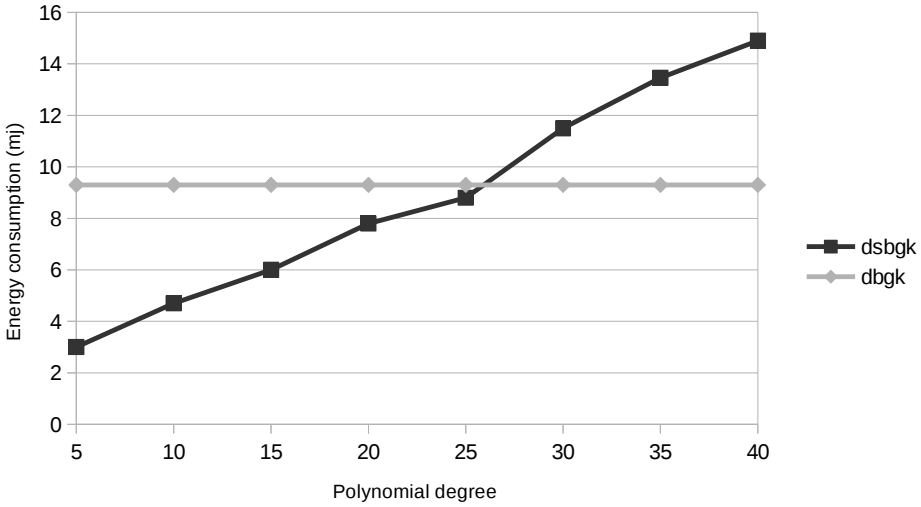

FIGURE 9
Setting polynomial degree

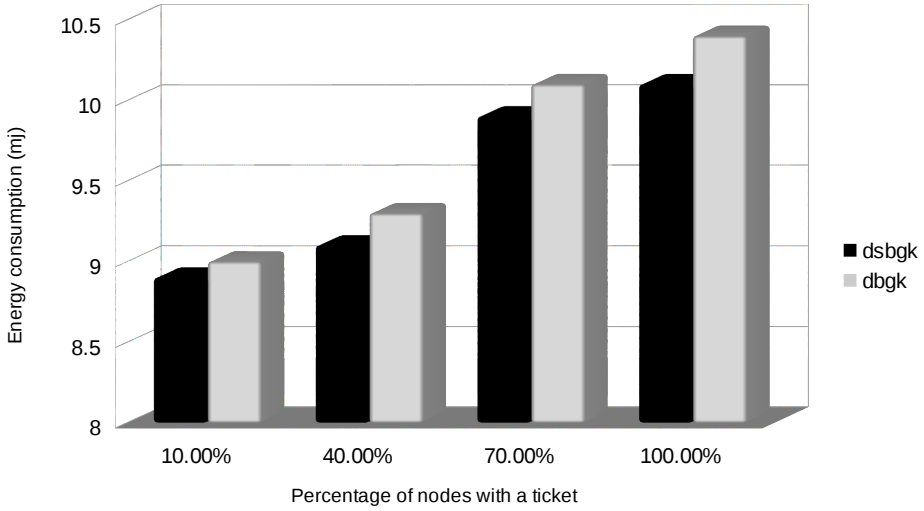

FIGURE 10
Member leaving cost comparison

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 [36] (previously mentioned in section 3.5), 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 [46]. 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 FIGURE. 10. 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 compared to the raise noticed in DBGK energy consumption. This is mainly due to the superior efficiency of polynomial computation compared to cryptographic symmetric primitives. As a result, we can safely affirm that no additional cost is induced by the obtained storage gains.

5.2 Second phase (DsBGK vs state of the art)

In this phase, we broadened our comparison by including other similar protocols from the literature (i.e. LKH[61], MARKS [15], and Veltri et al protocol [58]). We considered the following metrics:

- *Number of messages* exchanged between a member of the group and a controller (key management server for other protocols) following an event (typically a membership change).
- *Number of transmitted keys* contained in the exchanged messages as a result of an event.

Protocol	messages	keys to transmit
MARKS	1	$\leq 2(\log_2(n) - 1)$
LKH	h	h(h+1)/2-1
Veltri et al	1	$\leq 2(\log_2(n) - 1)$
DBGK	1	1
DsBGK	1	1

TABLE 3
Member joining (Initialization)

As mentioned in section 3.6, in case of a leave (or ejection), all members which have the same tickets as the leaving member need to be notified. As a result, the overall cost related to the number of messages $M1$ and the number of keys $M2$ can be considerable. To reduce this cost, the stat-of-the-art LKH [61] protocol can be superimposed to DsBGK. The long-term key SK would then be updated using LKH in replacement of a notification broadcast to the involved members.

In Veltri et al protocol [58], the group key is updated using LKH when a member leaves the group unpredictably. Due to the dynamic nature of collaborative groups, the unpredicted departure of members is likely to take place frequently. It is important to highlight that in DsBGK, LKH is only considered when the notification overhead is higher than LKH overhead. Consequently, we make sure that DsBGK cost is less than or equal to LKH cost [61] in the worst case.

The controller is in charge of choosing whether to use LKH or to use the notification process. It is worth noting that the cost of LKH regarding the number of messages to be sent is (d-1)(h-1). (d: the degree of LKH tree, h: height of the tree). The controller evaluates the cost of using LKH, as well as the cost of using a notification mechanism, and makes a decision regarding the use of the former or the latter one.

The performances of DsBGK are compared with the performances of MARKS[15], LKH [61], Veltri et al [58], and DBGK [7] protocols relying on the following set of events: joining, sending/receiving messages, member leaving with valid tickets, and member leaving without valid tickets. TABLE 3 to TABLE 6 depict the theoretical complexity of each protocol. Relying on these analytical results, we simulate the energy cost following group key updates to evalu-

Protocol	messages	keys to transmit
MARKS	0	0
LKH	0	0
Veltri et al	0	0
DBGK	1	Nslot
DsBGK	1	Nslot

TABLE 4
Triggering secured message exchanges

Protocol	messages	keys to transmit
MARKS	X	X
LKH	(d-1)(h-1)	(d-1)h(h-1)/2
Veltri et al	(d-1)(h-1)	(d-1)h(h-1)/2 + 1
DBGK	M1$\leq (d-1)(h-1)$	M2$\leq (d-1)h(h-1)/2$
DsBGK	M1$\leq (d-1)(h-1)$	M2$\leq (d-1)h(h-1)/2$

TABLE 5
Member leaving with a valid ticket

Protocol	messages	keys to transmit
MARKS	0	0
LKH	(d-1)(h-1)	(d-1)h(h-1)/2
Veltri et al	0	0
DBGK	0	0
DsBGK	0	0

TABLE 6
Member leaving without a valid ticket

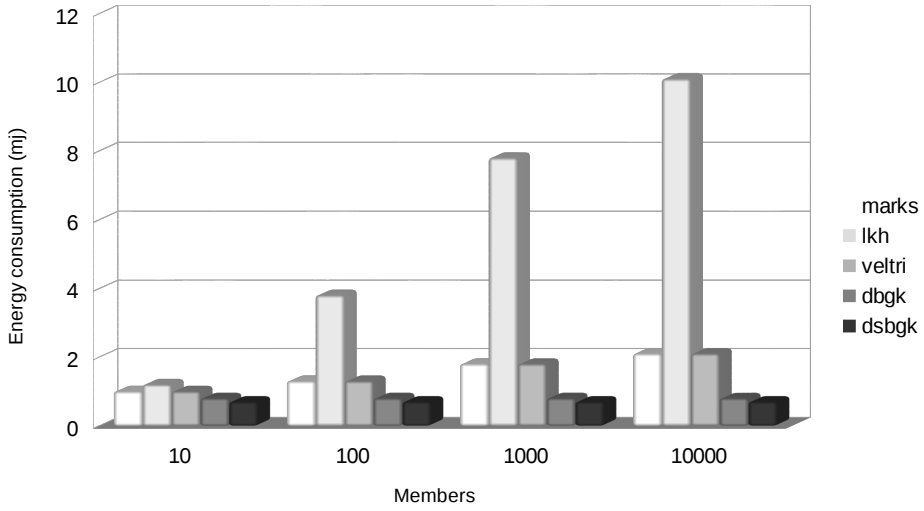

FIGURE 11
Joining: energy cost comparison

ate DsBGK performances using Cooja simulator compared to MARKS[15], LKH [61], Veltri et al [58], and DBGK [7] protocols.

FIGURE 11 shows the variation of energy cost following a join event in groups containing different numbers of users. We observe that LKH [61] energy consumption raises when the size of the group raises. This raise is more noticeable in comparison with MARKS[15] and Veltri et al [58]. Indeed, both Veltri et al [58] and MARKS[15] are time-driven protocols. These protocols only imply one message for the joining member. However, LKH [61] is an event-driven protocol. This protocol implies several messages in case of a joining event. DBGK [7], and DsBGK energy consumption is constant and independent from the number of users. It is worth mentioning that DsBGK energy consumption is slightly lower compared to DBGK [7].

In FIGURE 12, we depict the measured energy cost of a member leav-

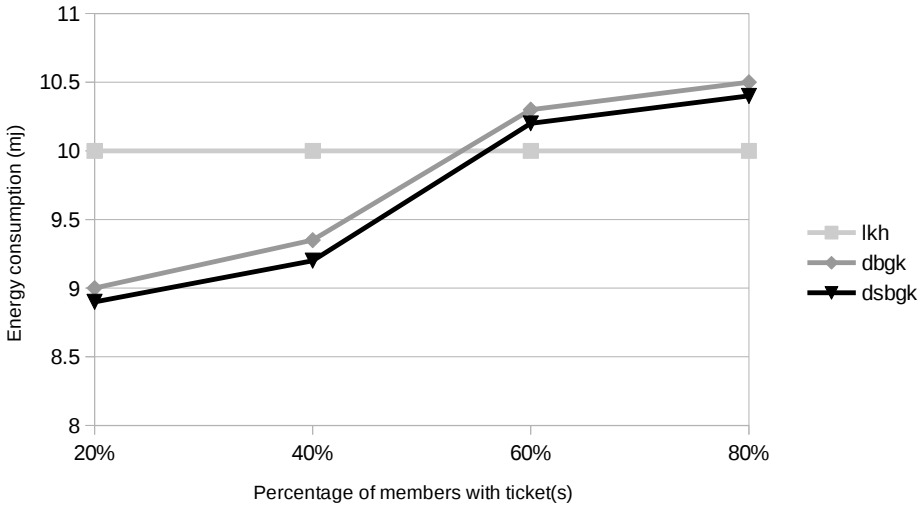

FIGURE 12
Leave: energy cost comparison

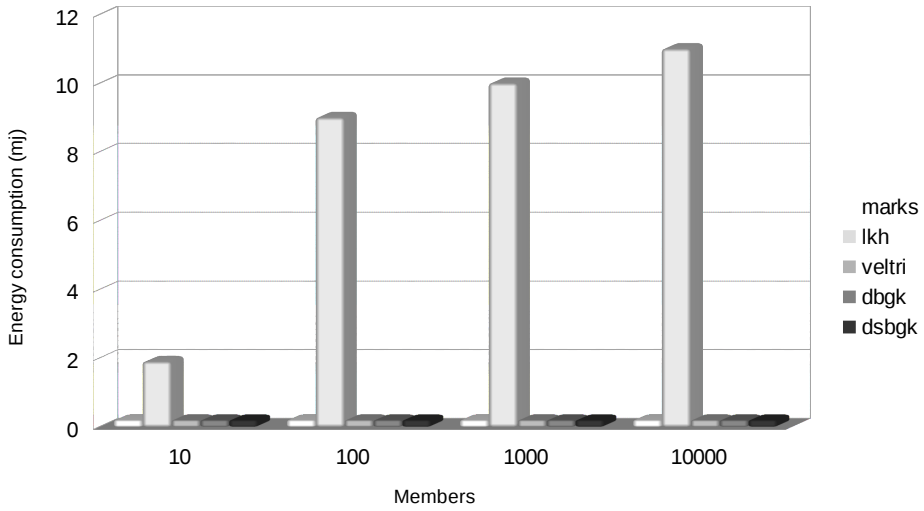

FIGURE 13
Leave (without a valid ticket): energy cost comparison

ing in possession of a valid ticket. We consider a group of 1000 users. We evaluate the energy cost while taking into account the proportion of members that have the same ticket as the leaving the member. Furthermore, we assume that the notified members will request 20 tickets (i.e. $NSlot$ = 20). In this experiment, MARKS is not considered as this protocol only treats planned departures. The results show that the energy consumption of LKH [61] (and Veltri et al [58] which relies on LKH for unpredicted leave events) is constant and it is not affected by the variation of the proportion of responding members. However, DBGK [7], and DsBGK energy consumption increases with the increase of responding members. It is worth noting again that compared to DBGK [7], DsBGK energy consumption is perceptibly inferior. We can notice that up to a proportion of around 50 % of members in the group in possession of a valid ticket at the moment of the departure, DBGK [7] and DsBGK energy consumption is lower than LKH [61].

FIGURE 13 depicts energy consumption after a leave event where the leaving member is not in possession of any valid tickets. The results show that LKH [61] energy consumption increases with the increase of the number of members. However, MARKS[15], Veltri et al [58], DBGK [7], and DsBGK perform the operation without any cost. This is due to the fact that LKH [61] is an event-driven protocol that is not based on time intervals. Thus, unlike other protocols, LKH [61] does not take any advantage of batch rekeying, which spare from updating the key if the leaving member is not in possession of valid credentials.

Unlike Veltri et al, LKH, and MARKS protocols that involve the totality of group members in rekeying operations, even if they have not been active for a long period of time prior to the rekeying, both DBGK [7] and DsBGK allow these members to remain in a sleep mode without being interrupted. Consequently, energy consumption is lower in up to a proportion of 50 % of group members in possession of the same tickets as those hold by the leaving member. This raise is caused by the computation and the communication overheads following an increasing number of messages to be dealt with. If the proportion exceeds 50 %, LKH is used for both DBGK and DsBGK. As a result, their energy consumption is less or equal to other protocols in the worst case. Furthermore, unlike Veltri et al, DBGK [7] and DsBGK are not based on predicting the leaving moment of the members. This property is particularly relevant, as IoT-based groups are likely to witness unpredictable events affecting their members. Indeed, DBGK [7] and DsBGK offer a flexible mechanism which allows members to ask for the exact number of keying materials that suits the most their storage capabilities (i.e. $NSlot$).

Following the security and performances comparison results, we can affirm that both DBGK and [7] DsBGK are suited to groups with a large number of members that can join the collaborative group for a long period of time (without being necessarily active), and leave the group unexpectedly. In addition, although DsBGK displays the same theoretical complexity as DBGK [7], simulation results show that DsBGK provides a perceptible energy gain due to the superior efficiency of polynomial computation compared to cryptographic symmetric primitives. Moreover, as presented in section 5.1, DsBGK preserves the same properties as DBGK [7], but provides a considerable improvement in terms of fault tolerance and scalability. The main source of scalability improvement is related to the fact that polynomial computation does not require shared credentials between constrained members and controllers. Thus, including additional controllers does not impact existing groups members. Moreover, it is worth highlighting that the improvement in polynomial building compared to both Patsakis et al [44] and Piao et al [45] schemes contributes as well to improve scalability. In fact, we introduced random values in the polynomial instead of adding salting parameters upon each member joining event, which would have drastically increase the polynomial degree and thus would have negatively impacted performances (weakening scalability). Furthermore, this randomness of polynomial terms also provides collusion freeness. Indeed, the disclosure of the private ID of colluding users brings no additional knowledge to retrieve private IDs of non-colluding members (see section 3.5).

Back to our e-health (i.e. PHR) use case scenario, and based on our evaluation studies, we can safely affirm that DsBGK is suitable to manage a group key that can be used to secure communication channels. Indeed, the role of controllers can be played by the group members, which are not constrained in terms of energy, computational, and memory resources, such as smartphones, personal computers, and servers. These controllers generate, and manage the group key on behalf of the constrained members of the group, such as e-health sensors. Backup controllers can be included in the group with very limited overhead on constrained members. Furthermore, no human intervention on the sensors is required, which might be very convenient, knowing that these sensors can be planted inside human bodies. Consequently, the failure of one or several controllers does not hamper the normal behaviour of the protocol, as backup controllers can take over. Besides, the improved efficiency in term of energy consumption for battery powered e-health sensors will increase their life time, and thus reduce the cycle of surgical interventions required for their replacement.

6 CONCLUSION

Resilient, scalable, and energy-aware group key management protocols constitute the cornerstone of modern distributed collaborative groups under the umbrella of the IoT paradigm. In this work, we built on top of an existing protocol (i.e. DBGK) to considerably enhance its features in terms of scalability, fault tolerance, and collusion freeness. To do so, we combined a polynomial based approach with DBGK to propose a new protocol called DsBGK. In fact, controllers secure their communications with constrained members using polynomials instead of symmetric encryption. As a result, there is no need for a manual intervention on the constrained members to store new credentials for each new controllers. Furthermore, we introduced an innovative application of blockchain technology to further enhance DsBGK strength in term of authentication. Security assessment including theoretical analysis, and formal validation proved the soundness of DsBGK security properties. Furthermore, simulation results showed that DsBGK improves both fault tolerance and scalability which are highly sought in sensitive applications, such as e-health systems. Compared to its peers from the state of the art, energy gains are achieved upon rekeying operations following membership events (e.g. join and leave). These results make DsBGK more suitable for heterogeneous, and dynamic collaborative groups. As a future work, we plan to design an actual implementation of DsBGK on real test-beds.

REFERENCES

[1] Avispa – a tool for automated validation of internet security protocols. *http://www.avispa-project.org*.

[2] The contiki operating system. *http://www.contiki-os.org*.

[3] Murphi model checker. *http://www.cs.utah.edu*.

[4] Prism - a probabilistic model checker. *http://www.prismmodelchecker.org*.

[5] Mohammed Riyadh Abdmeziem and Djamel Tandjaoui. (2014). A cooperative end to end key management scheme for e-health applications in the context of internet of things. In *International Conference on Ad-Hoc Networks and Wireless*, pages 35–46. Springer.

[6] Mohammed Riyadh Abdmeziem and Djamel Tandjaoui. (2015). An end-to-end secure key management protocol for e-health applications. *Computers & Electrical Engineering*, 44:184–197.

[7] Mohammed Riyadh Abdmeziem, Djamel Tandjaoui, and Imed Romdhani. (2015). A decentralized batch-based group key management protocol for mobile internet of things (dbgk). In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, pages 1109–1117. IEEE.

[8] Mohammed Riyadh Abdmeziem, Djamel Tandjaoui, and Imed Romdhani. (2016). Architecting the internet of things: state of the art. In *Robots and Sensor Clouds*, pages 55–75. Springer.

[9] Mohammed Riyadh Abdmeziem, Djamel Tandjaoui, and Imed Romdhani. (2016). A new distributed mikey mode to secure e-health applications. In *Proceedings of the International Conference on Internet of Things and Big Data - Volume 1: IoTBD,*, pages 88–95. SciTePress.

[10] Mohammed Riyadh Abdmeziem, Djamel Tandjaoui, and Imed Romdhani. (2017). Lightweighted and energy-aware mikey-ticket for e-health applications in the context of internet of things. *International Journal of Sensor Networks*, In press.

[11] Muhammad Salek Ali, Koustabh Dolui, and Fabio Antonelli. (2017). Iot data privacy via blockchains and ipfs. In *Proceedings of the Seventh International Conference on the Internet of Things*, page 14. ACM.

[12] Sedrati Anass, Abdelraheem Mohamed Ahmed, and Raza Shahid. (2017). Blockchain and iot: Mind the gap. In *4th EAI/Springer International Conference on Safety and Security in Internet of Things*. EAI.

[13] D. Balenson, D. McGrew, and A. Sherman. (February 1999). Key management for large dynamic groups: One-way function trees and amortized initialization. *Internet-Draft*.

[14] A. Ballardie. (May 1996). Scalable multicast key distribution. *RFC 1949*.

[15] B. Briscoe. (1999). Marks: Zero side effect multicast key management using arbitrarily revealed key sequences. *Networked Group Communication*, pages 301–320.

[16] Y. Challal and H. Seba. (2005). Group key management protocols: A novel taxonomy. *International Journal of Information Technology*, 2(1):105–118.

[17] A. Charu and T. Mathieu. (2009). Validating integrity for the ephemerizer's protocol with cl-atse. pages 21–32.

[18] Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, and and L. Vigneron S. Modersheim. (2004). A high level protocol specification language for industrial security sensitive protocols. *Proc. SAPS 04. Austrian Computer Society, 2004*.

[19] C. Chun, H. Daojing, C. Sammy, B. Jiajun, G. Yi, and F. Rong. (2011). Lightweight and provably secure user authentication with anonymity for the global mobility network. *International Journal of Communication Systems*, 24(3):347–362.

[20] EM. Clarke, O. Grumberg, and DA. Peled. (1999). Model checking. *MIT Press: Cambridge*.

[21] B. Daghighi, M.L.M. Kiah, S. Shamshirband, and M.H.U. Rehman. (2015). Toward secure group communication in wireless mobile environments: Issues, solutions, and challenges. *Journal of Network and Computer Applications*, 50:1–14.

[22] Roberto Di Pietro, Luigi V Mancini, and Sushil Jajodia. (2003). Providing secrecy in key management protocols for large wireless sensors networks. *Ad Hoc Networks*, 1(4):455–468.

[23] Gianluca Dini and Lanfranco Lopriore. (2015). Key propagation in wireless sensor networks. *Computers & Electrical Engineering*, 41:426–433.

[24] D. Dolev and C.C. Yao. (1981). On the security of public key protocols. *FOCS, IEEE, 1981*, page 350–357.

[25] J.R. Douceur. (2002). The sybil attack. *Peer-to-peer Systems*.

[26] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. (2011). Powertrace: Network-level power profiling for low-power wireless networks.

[27] Y. Glouche and T. Genet. (2006). Span – a security protocol animator for avispa – user manual. *http ://www.irisa.fr/lande/genet/span/, 2006*.

[28] Y. Hanna, H. Rajan, and W. Zhang. (2008). Slede: A domain specific verification framework for sensor network security protocol implementations. *Proceeding of the ACM Conference on Wireless Network Security (WiSec'08)*, pages 109–118.

[29] H. Harney and C. Muckenhirn. (July 1997). Group key management protocol (gkmp) architecture. *RFC 2093*.

[30] Abdel Alim Kamal. (2013). Cryptanalysis of a polynomial-based key management scheme for secure group communication. *IJ Network Security*, 15(1):68–70.

[31] Sye Loong Keoh, Sandeep S Kumar, and Hannes Tschofenig. (2014). Securing the internet of things: A standardization perspective. *IEEE Internet of Things Journal*, 1(3):265–275.

[32] Y. Kim, A. Perrig, and G. Tsudik. (2004). Tree-based group key agreement. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):60–96.

[33] A. Le, J. Loo, A. Lasebae, M. Aiash, and Y. Luo. (2012). 6lowpan: a study on qos security threats and countermeasures using intrusion detection system approach. *International Journal of Communication Systems*, 25(9).

[34] P. Lee, J. Lui, and D. Yau. (2006). Distributed collaborative key agreement and authentication protocols for dynamic peer groups. *Networking, IEEE/ACM Transactions on*, 14(2):263–276.

[35] S. Lim, T.H. Oh, Y.B Choi, and T. Lakshman. (February 2010). Security issues on wireless body area network for remote healthcare monitoring. *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), IEEE International Conference*, pages 327 – 332.

[36] Donggang Liu and Peng Ning. (2007). *Security for wireless sensor networks*, volume 28. Springer Science & Business Media.

[37] Niu Liu, Shaohua Tang, and Lingling Xu. (2013). Attacks and comments on several recently proposed key management schemes. *IACR Cryptology ePrint Archive*, 2013:100.

[38] M. Marino and U. Caterina. (2011). Formal analysis of facebook connect single sign-on authentication protocol. 11:22–28.

[39] Faiza Medjek, Djamel Tandjaoui, Mohammed Riyadh Abdmeziem, and Nabil Djedjig. (2015). Analytical evaluation of the impacts of sybil attacks against rpl under mobility. In *Programming and Systems (ISPS), 2015 12th International Symposium on*, pages 1–9. IEEE.

[40] Ralph C Merkle. (1987). A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 369–378. Springer.

[41] S. Mittra. (1997). Iolus: A framework for scalable secure multicasting. *ACM SIGCOMM Computer Communication Review*, 27(4):277–288.

[42] S. Moedersheim and P.H. Drielsma. (2003). Avispa project deliverable d6.2: Specification of the problems in the high-level specification language. http://www.avispa-project.org.

[43] Gérald Oster, Pascal Urso, Pascal Molli, and Abdessamad Imine. (2006). Data consistency for p2p collaborative editing. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 259–268. ACM.

[44] Constantinos Patsakis and Agusti Solanas. (2013). An efficient scheme for centralized group key management in collaborative environments. *IACR Cryptology ePrint Archive*, 2013:489.

[45] Y. Piao, J. Kim, U. Tariq, and M. Hong. (2013). Polynomial-based key management for secure intra-group and inter-group communication. *Computers & Mathematics with Applications*, 65(9):1300–1309.

[46] David A Plaisted. (1984). New np-hard and np-complete polynomial and integer divisibility problems. *Theoretical Computer Science*, 31(1-2):125–138.

[47] S. Rafaeli and D. Hutchison. (June 2002). Hydra: a decentralized group key management. *11th IEEE International WETICE: Enterprise Security Workshop*.

[48] S. Rafaeli and D. Hutchison. (2003). A survey of key management for secure group communication. *ACM Computing Surveys (CSUR)*, 35(3):309–329.

[49] S. Raza, L. Wallgren, and T. Voigt. (2013). Svelte: Real-time intrusion detection in the internet of things. *Ad hoc networks*, 11(8).

[50] Ronald L Rivest, Adi Shamir, and Leonard Adleman. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

[51] Antonio Ruiz-Martínez, C. Inmaculada Marín-López, Laura Baño-López, and AF Skarmeta. (2006). A new fair non-repudiation protocol for secure negotiation and contract signing. page 16.

[52] S. Setia, S. Koussih, S. Jajodia, and E. Harder. (2000). Kronos: A scalable group rekeying approach for secure multicast. *Proceedings IEEE Symposium on Security and Privacy*, pages 215–228.

[53] Adi Shamir. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.

[54] Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, and Alberto Coen-Porisini. (2015). Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164.

[55] Sabrina Sicari, Alessandra Rizzardi, Daniele Miorandi, and Alberto Coen-Porisini. (2016). Internet of things: Security in the keys. In *Proceedings of the 12th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 129–133. ACM.

[56] Paul C Tang, Joan S Ash, David W Bates, J Marc Overhage, and Daniel Z Sands. (2006). Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption. *Journal of the American Medical Informatics Association*, 13(2):121–126.

[57] L. Tobarra, D. Cazorla, F. Cuartero, G. Diaz, and E. Cambronero. (2009). Model checking wireless sensor network security protocols: Tinysec + leap + tinypk. *Telecommunication Systems*, 40(3-4):91–99.

[58] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari. (2013). A novel batch-based group key management protocol applied to the internet of things. *Ad Hoc Networks*, 11(8):2724–2737.

[59] Weichao Wang and Bharat Bhargava. (2005). Key distribution and update for secure intergroup multicast communication. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 43–52. ACM.

[60] Weichao Wang and Yu Wang. (2008). Secure group-based information sharing in mobile ad hoc networks. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 1695–1699. IEEE.

[61] C.K. Wong, M. Gouda, and S.S. Lam. (2000). Secure group communications using key graphs. *Networking, IEEE/ACM Transactions*, 8(1):16–30.