CAKE: An Efficient Group Key Management for Dynamic Groups

Peter Hillmann*, Marcus Knüpfer*, Tobias Guggemos†, Klement Streit*
*Universität der Bundeswehr München, †Ludwig-Maximilians-Universität München
Munich Network Management Team, Munich, GERMANY
Email: *{peter.hillmann, marcus.knuepfer, klement.streit}@unibw.de, †{guggemos}@nm.ifi.lmu.de

***Abstract*—With rapid increase of mobile computing and wireless network linkage, the information exchange between connected systems and within groups increases heavily. Exchanging confidential information within groups via unsecured communication channels is a high security threat. In order to prevent third parties from accessing this data, it is essential to encrypt it. For this purpose, the group participants need a common group key to enable encrypted broadcast messages. But efficient key management of secured group communication is a challenging task, if participants rely on low performance hardware and small bandwidth. Especially, dynamically changing group compositions generate large management overhead. For coordination and distribution, we present the modular group key management procedure CAKE that is centrally organized and meets strict security requirements. The lightweight G-IKEv2 protocol in combination with the key exchange concept of CAKE leads to an efficiently integrated solution. The hybrid approach combines the advantages of the existing protocols with the objective to reduce the computation and communication effort. It is shown that the procedure is more suitable for changing MANET groups than the existing ones. Moreover, the exchanged group key can be used for any services which provides a wide range of applications.**

***Keywords*—GKMP, IKEv2, wireless network, CAKE, LKH**

I. Introduction

In today's interconnected world, the wireless network linkage is growing rapidly. More and more devices are connected, especially in mobile and resource-constrained networks. Coming from fixed line networks, unicast communication have been predominant so far. During the last decade, we have witnessed the rise of new low power network technologies, such as ZigBee, IEET 802.15.4, Bluetooth, LoRa, SigFox, 4/5 G, just to name a few. Most of these have in common, that the bandwidth is limited and needs to be managed wisely, while the amount of information explodes with new scenarios in the context of IoT or Smart X. In order to meet the requirements in constrained environments, group communication is becoming more important for information exchange, as the benefits are reduced network overhead, computation power, and energy consumption. Efficiency is achieved by transmitting data packets only once, simultaneously to all group members. However, when it comes to security and privacy multicast lacks efficiency. Major challenges arise from the necessary management of the multicast group (in the following referred as communication group), which needs to be secured as well as the actual communication within the group.

In order to exchange confidential data within a group in a secure manner there exist methods to manage group keys, so called *Group Key Management Protocols* (GKMP). These protocols enable the secure access as well as the secure and efficient exchange of relevant information, such as group keys. Group keys are used for the encrypted communication within a group. Typically, all Group Members (GM) posses a symmetric key which is used to encrypt and decrypt the exchanged data. After all, it does not matter which service is using the GKMP infrastructure as underlying security technology.

Different areas of application do have distinct requirements to be covered by a group key management concept. Consequently, no existing concept could be applied universally to all areas. Depending on the dynamics of the group composition and changes, the calculation of cryptographic keys for the group communication, and the numbers of transmitted messages it could be demanding for the GMs concerning computation power, network bandwidth and energy consumption. As a consequence, existing solutions for secured group communication do not fulfill the requirements for mid-sized and large networks in resource-constrained environments. Exemplary use cases as in Figure 1, where an efficient and secure group communication is needed, are *Internet of Things* (IoT) networks, saftey-critical system networks, *Wireless Sensor Networks* (WSN), and *Mobile AdHoc Networks* (MANET) for public authority communication.

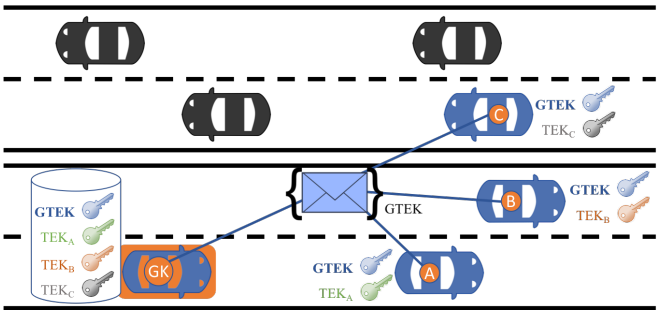

Figure 1: Secure and Efficient Group Communication in a Vehicular AdHoc Network (VANET)

The paper at hand proposes a mechanism to optimize the distribution and updating procedure of keys and thus aims on improving the general security properties in group communication in constrained environments. The contribution of this paper is the combination of an efficient networking protocol for group key management with a tree-based key update mechanism and a cryptographic key distribution scheme. The centralized *Group-IKEv2* (G-IKEv2 [3], [2]) protocol

is used for communication and initial key exchange with a key server. Thereon, the mechanism called *Central Authorized Key Extension* (CAKE [4]) is used to realize encrypted group communication. It utilizes the idea of *Secure Lock* [15] for compressing secured information into a single cipher and enhances its performance by the combination with a logical key hierarchy (LKH [19]) and new concepts for managing keys.

The remainder of this paper is structured as follows. In Section II a scenario illustrating the need of this hybrid approach as well as the requirements is given. Section III introduces related work on secure group communication and evaluates security as well as efficiency regarding the given scenario with a special focus on the *Logical Key Hierarchy* (LKH). Afterwards, Section IV describes the concept of our hybrid system combining the light-weight G-IKEv2 protocol and *Central Authorized Key Extension*, called CAKE. Finally, Section V evaluates the findings and compares with the well-established LKH, before Section VI summarizes and concludes this paper.

II. Scenario and Requirements

This work aims at concepting a highly efficient, but secure group key management scheme to facilitate secure group communication. The motivation is mainly found by the manifold use of constrained devices in a wide range of applications such as civil, industrial or military use cases. A prominent example for a civil application is car-to-car communication (see Figure 1), mainly revealing highly dynamic group formations and wireless network limitations. In contrast, in home automation scenarios (think of smoke detectors on battery) or military applications such as head-mounted units limitations in terms of availability of power, main memoy and storage, CPU and network datarates prevail.

Encrypted communication among a set of more than two group members is common to all scenarios. Thus, it seems desirable to share one cryptographic key among the group members in order to encrypt message transfers. Unfortunately, the management of such a key becomes costly quickly due to the dynamics withing a group (i. e. members joining or leaving the group rather frequently). Changing the cryptographic material upon every single group management action seems unavoidable, which motivates working towards other than naive approaches. Analyzing diverse application areas leads to a set of requirements that can be organized into mostly two categories – security requirements and scenario-driven non-functional requirements.

Security requirements

Forward Secrecy: Whenever a group member leaves the group or is expelled, the member in question must not be able to have access to a valid group key.

Backward Secrecy: Whenever a new group member joins a group, the member in question must not be able to have access to a formerly valid group key before joining.

Key Independence: Having access to one key must not yield the possibility to deduce another member's key.

Collision Free: Additionally, specific to group communication, there must not be a subset of group members that can deduce another member's key(s) by combining their knowledge.

Minimal Trust: A certain trust relationship will be found mandatory, but it shall be subject to minimize.

"CIA:" – Confidentiality, Integrity and Authenticity must be granted any time, also implying the avoidance of man-in-the-middle attacks or data injections.

Scenario-driven non-functional requirements

Low Datarates: The amount of transmitted data shall be minimal in order to facilitate network limited applications.

No 1-to-n Effect: Limited impact of a single membership change on all the other group members is mandatory, meaning not suffering from the 1-affects-n phenomenon, if a single membership change in the group affects all the other group members.

Minimal Delay: The delay imposed by the use of both management actions and cryptographic operations must be minimal.

Minimal amount of key changes and exchanges: The amount of management actions such as exchanging (new) keys shall be limited to a necessary minimum (not implying anything about the total amount of keys in general).

Low calculation complexity: Especially in scenarios exposing CPU limitations, a low complexity of cryptographic calculations is vital, while keeping up the maximally possible security level at the same time.

Compatibility: Clients not capable or not willing to support fancy optimizations should not be excluded from the communication. This is why a potential fall-back to simple (and standardized) mechanism should be supported.

Scenario-driven functional requirements

Based on the scenario, this yield in the following requirements for the group operations to be supported while complying the security requirements:

Join: One or more participants accede to an existing group. (Backward Secrecy)

Leave: One or more group members quit the group membership. (Forward Secrecy)

Re-Keying: Updateing the group key using an efficient procedure. (Prevent statistical analysis)

Merge: A common key can be efficiently provided to several groups by re-keying. (Backward Secrecy)

Split: A group is divided into several subgroups. (Forward Secrecy)

III. Related Work

Secure group communication is an extensively studied area and resulted in a couple of standardization activities (most recently a new standardization group for group key distribution was formed within the IETF[1]).

Rafaeli et al. [5] survey a set of approaches for secure group key distribution (GKD). According to their analysis, there are three different types of GKDs: centralized, decentralized and distributed GKD protocols. Most of the protocols considered

[1] https://datatracker.ietf.org/wg/mls - Started in February 2018

are rather *Cryptographic Key Schemes (CKS)* than networking protocols, but some of them are included in *Group Key Management Protocols*. The paper at hand offers the integration of an optimized *Cryptographic Key Schemes* into a centralized management protocol. Thus, the following section is divided in *Group Key Management Protocols* for communication and *Cryptographic Key Schemes* to manage the group key. To our knowledge, none of the approaches provides an efficient and integrated solution, especially with focus on low resource requirements. This is one of the reasons why the *Internet Engineering Task Force* (IETF) started a standardization process for group key distribution in February 2018 [6].

A. Group Key Management Protocols

A high-level definition of *Group Key Management Protocols* (GKMP) and their corresponding architecture is given by the IETF standard body in RFC 2093 [7] and RFC 2094 [8]. The development of actual GKMPs builds on top of these specifications and usually goes hand in hand with the development of a peer-to-peer key exchange protocol and its corresponding architecture. The *Internet Security Association and Key Management Protocol* (ISAKMP, RFC 2408 [9]), and *Group Domain of Interpretation* (GDOI, RFC 6407 [10]) have been the first instantiations. The requirements and design of these protocols were derived from multicast architectures of network vendors. Both, peer-to-peer key exchange and Group Key Management were revised for the sake of stronger security properties and better performance, resulting in *Internet Key Exchange v2* (IKEv2, RFC 7296 [3]) and the currently proposed G-IKEv2 [2] for groups. As G-IKEv2 offers a reduced networking overhead and includes a structure for distributing hierarchical keying information, we build the design of our solution on top of G-IKEv2 and the latest architecture from RFC 4046 [11].

B. Cryptographic Key Schemes

Centralized Cryptographic Key Schemes (CKS) comprise a central control authority to manage the group key and to coordinate the cryptographic procedures, often based on a GKMP. In contrast, decentralized techniques share the management of the keys between several instances [12], [13]. Thereby, the generation and distribution of group keys is realized by cooperative instances, which are typical hierarchically ordered. In addition, distributed key agreement procedures delegate the key generation process to not only an individual group member, but to a group of members.

One example is the *Group-Diffie-Hellman* Key Exchange [14], but others exist [5]. All members of a group are organized in a virtual topology, typically into a ring, hierarchies on basis of trees, or just unstructured. In all these schemes, every member of a group shares a common *Transport-Encryption-Key* (TEK).

Another approach is dividing groups into subgroup with individual TEKs. A master within every subgroup takes care of the communication and keys, which allows avoiding 1-to-n effects while re-keying [12]. The downside is requiring repetitive conversions of encrypted messages between the subgroups. Within the subgroups, these approaches use key management techniques of the three shown categories why out of scope of this work.

Despite their structured nature, centralized CKS can further be categorized into one of the three subcategories:

- Pairwise keys: Transmission of the group key by the central instance via individual subscriber communication
- Broadcast secret: Transmission of the group key via broadcast instead of individual secured connections
- Hierarchical structure: Coordination of participants in a tree structure with corresponding cryptographic subkeys

The first and most widely recognized CKS ever is defined in the GKMP, which belongs to the category of the pairwise keys. The central server shares an individual secret key with each group member, which is called the *Key-Encryption-Key* (KEK). For a common TEK of a group, the server generates these. Subsequently, the server sends the group key to each participant individually encrypted using the KEK. Upon change of the group constellation, the entire group is re-created, leading to high management and communication overheads.

An example for the broadcast secret is the *Secure Lock* (SL) [15], [16] that enables the creation of a group or a re-keying action using a single broadcast message. The SL scheme is based on the *Chinese Remainder Theorem* (CRT) [17], [18], which uses the properties of congruence to encrypt. However, the reduction of communication overhead is obtained by more complex calculations compared to GKMP so that this approach only renders feasible in special scenarios.

A compromise are schemes building on hierarchical structure. A well-known approach is *Logical-Key-Hierarchy* (LKH) [19], [20], which is integrated into GDOI and G-IKEv2. The KEK's and the group participants are maintained in a binary tree. Each node in the tree represents a KEK that is known to the underlying nodes. Maintaining the associated keys of the tree structure increases the management effort, especially the calculation and distribution of internal keys. This approach offers a moderate advantage only in case of repetitive leavings of group members. Since this operation does not take place in every secured group, this is unnecessary effort.

Focusing on the motivation for this paper, a centralized scheme with common TEK renders mandatory, especially in order to control and authorize individual members of a group. In this paper, a combination of the advantages of GKMP, SL and LKH as CAKE [21] with an integration into G-IKEv2 is proposed, allowing for efficient key management.

IV. Concept

Targeting highly efficient and encrypted group communication, this paper proposes the combination of lightweight *G-IKEv2* ([2]) for the key exchange and *Central Authorized Key Extension* (CAKE) [4] for the group key management. CAKE's key management is centrally organized and requires a trustworthy *Group Controller* (GC). The GC is responsible for the generation, administration and distribution of the keys and thus requires more computational power than any other (lightweight) group member.

The remainder of this section is organized into subsections inspired by group management operations and patterns:

(A) Client-Server communication based on G-IKEv2
(B) Member Registration on the GC
(C) Group and Group Key Creation
(D) Re-Key of the group
(E) Join of member(s) to a secured group
(F) Leave / Exclude of member(s) from a secured group
(G) Tree Management and Key Addressing
(H) Merging and splitting groups

A. Client-Server communication based on G-IKEv2

G-IKEv2 [2] is used to secure the transmission of cryptographic material for CAKE as it has already proven suitable for constrained devices [1]. G-IKEv2 already supports the establishment of a confidential and authenticated 1-to-1 channel between a client and the GC. It also offers the distribution of *Group Transmission Encryption Key*s (GTEK) and *Group Key Encryption Key*s (GKEK) and thus only requires additional support for CAKE. To communicate securely in a group, every group member has to possess a GTEK used for the communication in the group and a GKEK used to distribute the GTEKs securely. Figure 2 gives an overview about G-IKEv2:

1) **Key Exchange**: A G-IKEv2 key exchange can be divided into two phases:
   a) Establishing an *Initial Security Association* (IKE_SA_INIT): The first two messages from the client to the GC and back establish a *Security Association* (SA) and thus a secure channel between the client and the server (Phase ①: Initialization).
   b) Exchanging keys (GSA_AUTH): Given the secured communication path, the client identifies and authenticates itself and in turn receives transport and key encryption keys (GTEK and GKEK) from the server. The *Group Security Association (GSA) Policy* includes the security parameters (algorithms, lifetime, etc.), while the actual keys are transported within the *Key Download (KD) Payload* (Phase ②: Group Lifetime).
2) **Re-Keying** (GSA_REKEY): Whenever a GTEK or GKEK loses validity (e. g. being outdated), a re-keying action is triggered by the server (GSA_REKEY), which is close to equal to the GSA_AUTH phase (Phase ③: Group Key Refresh).

B. Member Registration on the GC

Each participant P_i registers with CAKE by negotiating an individual key pair (Key_i) with the GC during an IKE_SA_INIT exchange (①). The initial exchange is done with a Diffie-Hellman key exchange, which by design lacks authenticity. A second message GSA_AUTH (②) is used to authenticate both, the client and the GC. Note, that the GSA_AUTH can be used to directly join a group as part of the registration process (see Section IV-E).

C. Group and Group Key Creation

On request, the GC randomly generates a GTEK and GKEK. According to G-IKEv2, the GC manages cryptographic material and algorithms for every group. They are stored in the *TEK_SA* and *KEK_SA* databases (see Figure 2).

The GC may decide to create a new group with the new group key and members already registered and authenticated by building a GSA_REKEY payload as follows:

1) The GC constructs a CRT congruency in analogy to the SL scheme, so-called *Lock MX*. Therefore, it uses the individual m_i and Key_i from all participants of the specified group to calculate the Lock MX to encrypt the GKEK (see CRT calculation [17], [18]).
2) The GTEK is encrypted with the GKEK. For the sake of efficiency and security [22], XOR-operations are used for bitwise encryption of the new key tuple with a hashed GKEK. However, any encryption method specified by G-IKEv2 is supported.
3) The keys are embedded into a `CAKE_PRIME` GSA Policy (including the new `KEK_MANAGEMENT_ALGORITHM` called `CAKE`) and a `CAKE_PRIME` KD payload. They are distributed using a single GSA_REKEY broadcast message.

A participant P_i can only "open" the Lock MX, if she possesses a value m_i that was included during the creation of the lock. In consequence, only intended recipients (i. e. group members) are able to read the GKEK and GTEK by solving the CRT.

D. Re-Key of the group

In case the $GTEK$ needs to be renewed, a re-keying action is carried out. The GC generates the keys $GKEK_{new}$ and $GTEK_{new}$, which will be encrypted using the $GKEK_{current}$, embedded into the KD and broadcasted with a GSA_REKEY message. In order to grant forward and backward secrecy, a re-keying action is also carried out every time a member joins or leaves the group.

E. Join of new member(s) to a secured group

If a new participant P_{i+1} wishes to join the group, she sends a GSA_AUTH request including the group ID Id_g she wishes to join. The GC authenticates P_{i+1} and generates an inhomogeneous prime number m_{i+1} for a CRT congruency for P_{i+1}. Additionally, a new GSA policy and KD payload called `CAKE_PRIME` is added, holding m_{i+1}. The use of CAKE is communicated with a new `KEK_MANAGEMENT_ALGORITHM` called `CAKE` within the GSA Policy (see Section 4.5.1.1 in [2]). The GC also generates $GKEK_{new}$ and $GTEK_{new}$ and embeds the information and keys into an GSA_AUTH sent to the new group member via unicast.

Additionally, a re-key is triggered for any of the former group members. The re-key includes the KD $(GKEK_{new}, GTEK_{new})$ encrypted with the $GKEK_{Current}$. Switching from $GTEK_{old}$ to $GTEK_{new}$ enables the enlarged group (former group plus joined members) to communicate securely. As long as the $GTEK_{old}$ and $GKEK_{old}$ are still secure the $GTEK_{new}$ and $GKEK_{new}$ should be generated by hashing the *old* once. So the keys do not need to be distributed over the network. Only a tiny information message is necessary.

A mass entry of more than one new participant is equivalent to the process as described before, whereas the $GTEK_{new}$ is send to every new member individually. Alternatively, the new participants can be combined together via a CRT to

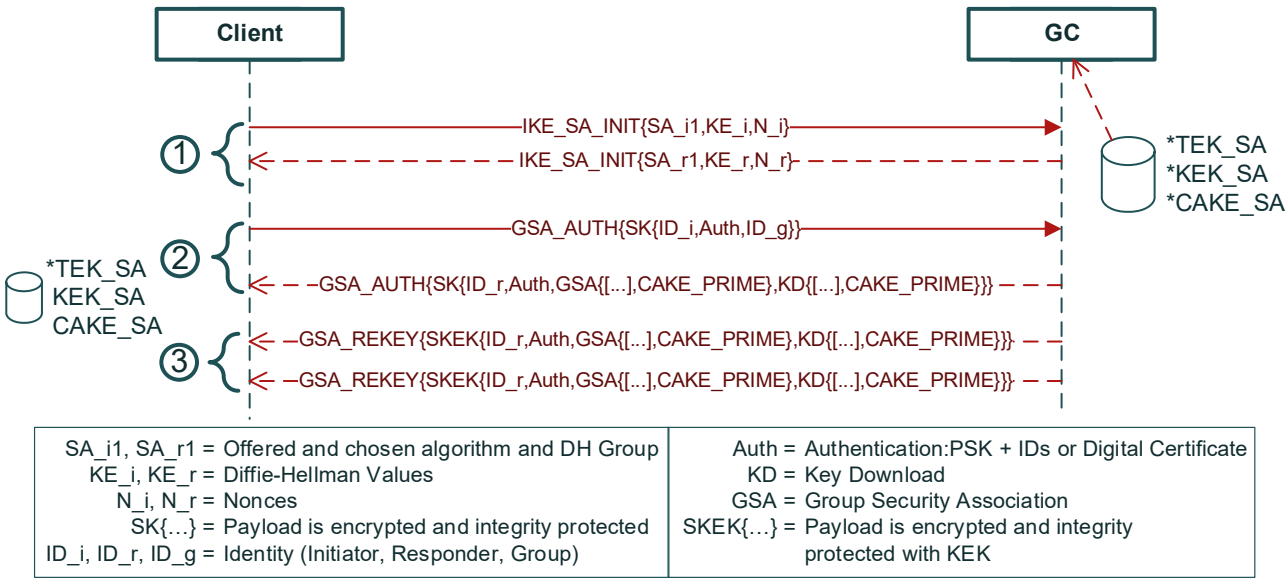

Figure 2: G-IKEv2 exchange with CAKE features

transmit the $GKEK_{new}$ so that only one message is necessary instead of multiple individual ones. Unfortunately, the latter is only possible if the joining clients are already authenticated. In both cases, two GSA_REKEY messages are broadcasted, one holding the Lock MX for the new clients and one with $(GKEK_{new}, GTEK_{new})$ encrypted with the $GKEK_{Current}$ for the former group members. Thus, an arbitrary number of new members joining a group requires a constant number of messages and thus scales efficiently with the amount of new members.

F. Leave/Exclude of a member from a secured group

Withdrawal of a member from a group can be initiated by the participant herself or be determined by the GC as exclusion. In any case, the presently known $GTEK_{Current}$ and $GKEK_{Current}$ cannot be used, as the expelled participant is in possession of them. To reduce the effort, CAKE uses a reduced CRT system and a ternary tree structure, which is managed by the GC.

Figure 3 illustrates CAKE's tree structure with level A (the root) representing the GKEK and GTEK. Every node represents a pair of keys (m_t and key_t) known by the underlying participants. The actual group members with their personal secrets m_i and key_i are mapped to the leaf nodes of the tree. The designation mX of a node defines a specific m_i for the CRT system.

All pair of keys on the path from the root to the participant must be known by the participant. The tree structure enables efficiency, but its creation can be deferred and only be initialized if necessary. This allows the tree being set up and distributed during a period of low network load. Considering the state of the art, nearly any tree-based scheme ignores this issue and excludes the costs for the tree setup in the evaluation.

Due to their flat structure, trees with more than two subnodes are better suited for larger groups than binary trees. In most scenarios (rarely more than 60 participants and hard to imagine more than 300 [23]), the ternary tree structure is ideal with regard to the size of the tree.

G. Tree Management and Key Addressing

In order to differentiate between different keys, an efficient addressing scheme is mandatory. This applies to every key, not just the keys in the tree. The ID of every key pair is defined as an $8x2$ bits address. This address space allows a maximum tree depth of 8 and 2,187 group members in a group with 3,280 key pairs in the tree. The IDs are starting from `00` at the root key pair on the top. Every parent has the children `01`, `10` and `11`. The unused bits are padded with `00`. This allows a unique identification of the position of every key within the tree.

Temporary keys can be derived easily like the keys for re-keying actions. These key pairs not yet included in the tree structure obtain the ID starting with `11`. After a client is authenticated with the GC, it has its own secret m_i and key_i (see Section IV-E), distributed with an address within or outside the tree. This support the re-balancing if the tree if necessary by the GC.

In order to take full advantage, the CAKE protocol for the distribution of keys requires the implementation of the following messages: I.) Downloading key pairs on the path to the root from the GC - *CAKE_Download_Array* II.) Re-Addressing of keys - *CAKE_Update_Array* III.) Receiving updates of key pairs - *CAKE_Readress_Array* IV.) Re-Keying upon removal of group members - *CAKE_Leave_Array*

This is similar to the payloads already defined in G-IKEv2 for *LKH Download Type* (see Section 4.8.3 in [2]). According to this, these four CAKE Download Types and three substructures are defined as array elements. This support the compatibility. As shown above, any message can be distributed securely as broadcast. Furthermore, the information of all download types are structured as an array to differentiate between the elements. Each elements of the arrays is indexed with the identity of the key pair in the tree. Using the unique

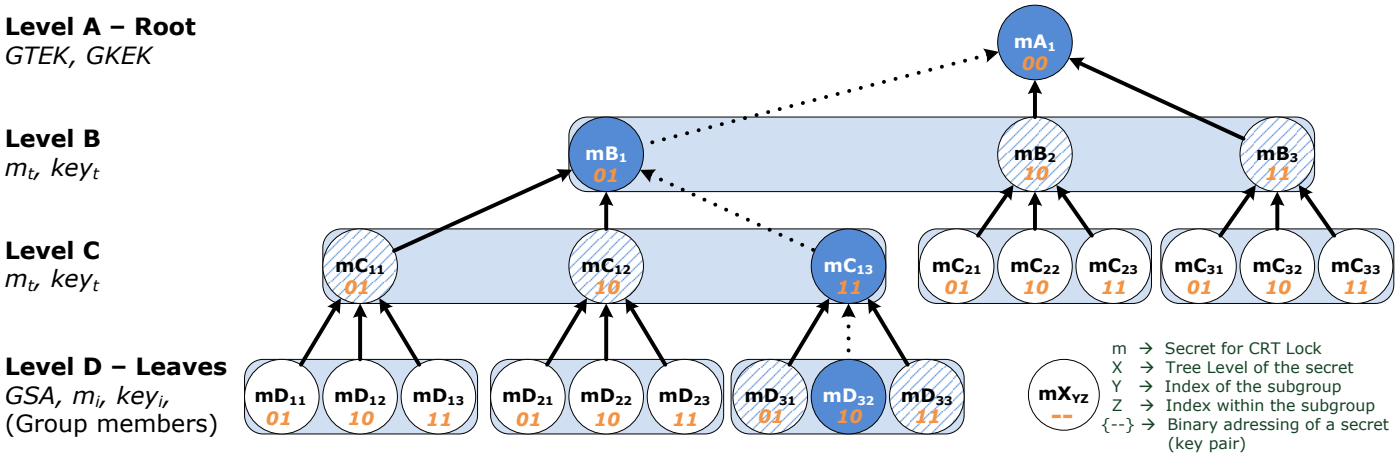

Figure 3: Ternary tree structure to manage the keys and to reduce the calculation effort by withdrawal.

```
 0        1        2        3
+--------+--------+--------+--------+
|   CAKE Key Id   |     RESERVED    |
+--------+--------+--------+--------+
| Size CRT key_id |    Size m_id    |
+--------+--------+--------+--------+
~            CRT (key_id)           ~
+--------+--------+--------+--------+
~      m_id (encr with key_id)      ~
+--------+--------+--------+--------+
```

(a) CAKE Keys Substructure

```
 0        1        2        3
+--------+--------+--------+--------+
|      # Keys     |     # Leaves    |
+--------+--------+--------+--------+
~    CAKE_Leaves_Substructure []    ~
+--------+--------+--------+--------+
~     CAKE_Keys_Substructure []     ~
+--------+--------+--------+--------+
```

(b) CAKE Leave Array

Figure 4: Exemplary CAKE Structures for G-IKEv2

tree IDs, the nodes can detect if they are affected by the action or not by comparing address prefixes.

The *CAKE_Download_Array* can be embedded in GSA_AUTH or GSA_REKEY message. It holds multiple keys, transported within a Key-Substructure (see Figure 4a). Therefor, a CRT for m_i and key_i is built using the key pairs of the child nodes in the tree. The substructure includes the address of the key pair and the CRTs themselves. For efficiency, the tree has to be distributed bottom up as other tree based group key management schemes. If a client is in possession of one key pair, then he is able to solve the Lock MX to obtain the information.

For the re-addressing of the keys, we use the *CAKE_Update_Array* embedded in a GSA_REKEY message. Due to the message type, the receiver knows, that there have been distributed keys before. Thus, these keys need to be updated in the client's CAKE_SA.

To change the ID of a key pair, the *CAKE_Readress_Array* is used. With this message, a key pair obtains a new position in the tree. Furthermore, it can be used to include nodes to the tree or to re-arrange subtrees (e. g. when the tree is re-balanced). The readdress information is holding in tuples (Id_{old}, Id_{new}). If a key on his path to the root receives re-addressing, it needs to update all keys on its path further down the tree.

When a node leaves or has to be excluded, a new root key pair has to be created. It is used to encrypt the GTEK which is embedded in a KD Payload. This information is communicated with the *CAKE_Leave_Array* message. A leaving node is in possession of all key pairs on the path to the root (see Figure 3: node mD32, dark, binary address: 00-01-11-10). All these marked key pairs can not be used for further security operations. Instead, all mX located next to a marked node on the same level are used (see hatched nodes in Figure 3). This is done for the entire tree along the path to all key pairs, the leaving client is in possession of. The updated key pairs are embedded in a Keys-Substructure (see Figure 4a) which in turn is embedded in the *CAKE_LEAVE_ARRAY* (see Figure 4b). The operation allows excluding multiple nodes with only one message.

Please note that the GC may choose to distribute only the root key pair with a short and tiny message. The other keys on the path are updated later with *CAKE_Update_Array* instead of one large *CAKE_Leave_Array*.

H. Merging and splitting groups

To merge two or more existing groups, it requires a renewal of several keys. The new common $GTEK_{new}$ will be spread based on the currently used individual GKEKs of the merging groups. It is mandatory to send one message per group to create a merged group.

A group split is done by re-addressing the sub-keys within the key tree and building Lock MX in the amount of divivded subgroups including the new keys. The number of messages may be as small as one, but depends heavily on the previous tree structure. As every leave operation, it is coherent with a high effort and will be analyzed in detail in further studies.

Please note that the usage of the *Delete Payload* message as specified in [2] is not resistant to malicious attacks of internal group members. In the cryptographic community, this problem is referred to as Post Compromise Security, which is an unresolved unresolved problem. So CAKE avoids the usage of this message and the process has to be authorised through the group controller.

V. Evaluation

Having a sound concept at hand, this section evaluates CAKE under the following three aspects: Firstly, a (theoretical)

Table I: Comparison of CAKE with LKH and traditional GKMP (represented by G-IKEv2), with special regards on cryptographic overhead. The keys are defined for AES with 16 Byte keys.

	Register/Join	**Mass Join**[2]	**Key Download/Update**[2]	**Tree Operation**	**Leave**[2]
Networking (in Bytes)	KD Payload[1]: 12 Byte				
GKMP	KEK: 16 TEK: 16 Key: 16	p Messages with: KEK: 16 TEK: 16	n Messages with: KEK: 16 TEK: 16	*undefined*	$n-1$ Messages with: KEK: 16 TEK: 16
LKH	KEK: 16 TEK: 16 Key: 16	p Messages with: KEK: 16 TEK: 16	1 Message with[3]: Hdr: $(4+(n-1)*12)$ Keys: $(n-1)*16$	*same as Key Download*	1 Message with[3]: Hdr: $4+log_2(n)*8+\sum_{i=1}^{log_2(n)-1} i*8$ Keys: $\sum_{i=1}^{log_2(n)-1} i*16$
CAKE	KEK: 16 TEK: 16 Prime: 17 Key: 16	1 Message with: Hdr: 16 TEK: 16 KEK[5]:$\lvert CRT(p)\rvert$	1 Message with[3]: Hdr: $4+n*12$ Keys[5]: $(log_3(n)-1)*\lvert CRT(3)\rvert$ Primes: $(log_3(n)-1)*3*17$	1 Message with[3]: Hdr: $(4+8)$ Key[5]: $\lvert CRT(3)\rvert$ Primes: $3*17$	1 Message with: Hdr: $12+log_3(n^2)*8$ TEK: 16 KEK[5]: $\lvert CRT(log_3(n^2))\rvert$
Computation					
GKMP[4,7]	GC: $O_K(1)$ Cl: $O_K(1)$	GC: $O_K(p)$ Cl: $O_K(1)$	GC: $O_K(n)$ Cl: $O_K(1)$	*undefined*	GC: $O_K(n-1)$ Cl: $O_K(1)$
LKH[4,7]	*see GKMP*	GC: $O_K(p)$ Cl: $O_K(1)$	GC: $O_K(2^{log_2(n)+1})$ Cl: $O_K(log_2(n)+1)$	*same as Key Download*	GC: $O_K((log_2(n)+log_2(n-1)))$ Cl: $O_K(log_2(n))$
CAKE[6,7]	*see GKMP*	GC: $O_L(p)$ GC: $O_K(1)$ Cl: $O_L(p)$ Cl: $O_K(1)$	GC: $O_L(3*\frac{n-1}{2})$ GC: $O_K(1)$ Cl: $O_L(3*log_3(n))$ Cl: $O_K(1)$	GC: $O_L(3)$ GC: $O_K(1)$ Cl: $O_L(3)$ Cl: $O_K(1)$	GC: $O_L(log_3(n^2))$ GC: $O_K(1)$ Cl: $O_L(log_3(n^2))$ Cl: $O_K(1)$

[1] Required for every Key distributed with G-IKEv2 [2] n being Group Members, p number of members joining or leaving [3] KEK and TEK is carried as in GKMP
[4] GC performs *encrypt* and Client performs *decrypt* [5] $\lvert CRT(i)\rvert$: size of CRT with i elements in Bytes [6] O_L: Complexity of creating/solving Lock MX.
[7] O_K: Complexity of encryption/decryption of keys.

comparison of the computational complexity as well as networking load of CAKE, LKH and traditional GKMPs is carried out. Secondly, an implementation of CAKE for RIOT OS proofs both its lightweight nature and its applicability in constrained scenarios. Given the result, the section will close by evaluating CAKE against the requirements as stated in Section II.

A. Comparison with LKH and GKMP

Existing concepts for managing group keys are traditional GKMP systems (represented by G-IKEv2) and LKH. Table I contains the comparison of CAKE with these two concepts regarding the networking and computation overhead. On the one side, it is indisputable that as no client leaves the group the traditional GKMP approach performs optimal. On the other side, LKH and CAKE perform far better in terms of quantity of messages and computations when *Forward Secrecy* is required at the moment clients leave. This is a major benefit of these two concepts.

Although CAKE requires a pair of keys (m_i and key_i) to be sent when distributing the tree, it can outperform the LKH mechanism introduced in G-IKEv2. The amount of key headers is equal in both systems. Unfortunately, LKH tree entries need to be transported multiple times decreasing its efficiency. Using a CRT system, CAKE offers the distribution of keys using a single message. Beside this, the message can be send to a later point of time, when network load is low. Although, the size of the resulting Lock MX increases linearly (see Table II), it still decreases the necessary protocol information heavily.

CAKE also reduces the demand for computational power on the client-side. Instead of carrying out multiple decryption operations (as for example LKH would do), the client has to perform one single modulo and one decrypt operation only. Nonetheless, this comes at the price of storing more cryptographic material (m_i, key_i) compared to LKH where only key_i has to be stored for every node in the tree.

1) Network overhead of LKH: As LKH operates very similar as CAKE, the following will comparing both overheads. LKH uses a binary tree and its G-IKEv2 extension can currently handle a maximum number of 65,536 participants. For comparison we assume a tree with 11 levels resulting in a maximum of 2,048 participants. Removing one client from the key will therefore result in 10 keys having to be changed and distributed in the network (for better insights to LKH in G-IKEv2 we recommend Appendix A of [2]). This would result in 10 LKH_UPDATE_ARRAYs carrying a total of $\sum_{i=1}^{10} 16*i+8*i+8 = 1{,}400$ Bytes (16, being the Key size, 8 being the LKH Keys header and 8 being the LKH_UPDATE_ARRAY header). Please note that in the current version of the G-IKEv2 extension for LKH, many keys are transported multiple times which heavily decreases efficiency. With some optimizations, the LKH Key Download could be decreased to 464 Bytes and, thus, being slightly more efficient than CAKE in terms of networking. However, this slightly better efficiency comes with the cost of higher computation overhead on the client, as in the worst case, 10 keys have to be decrypted individually. Additionally, changing the tree can currently not benefit from an address scheme as proposed in CAKE, making this operation more expensive in terms of networking and computation.

2) Unoptimized re-key networking overhead: Using the G-IKEv2 protocol without any optimizations (such as CAKE or LKH) would result in number of participants messages including 80 Bytes overhead for KEK and TEK. Assuming the current maximum of 2,187 participants, this would result in 2,187 messages. The IETF draft for G-IKEv2 [2] defines an additional GSA_INBAND_REKEY message for such tasks. Even without optimization, the new keys could be carried out in a single broadcast message, encrypting the KEK with every privately shared secret between client and server. The GSA_REKEY message would carry one KEK Key Download Types (40 Bytes) for every participant and one TEK Key

Table II: Required time for Lock MX operations with i elements. For comparison, the time to encrypt and decrypt the key hierarchy of LKH with tree depth i is shown. The number of clients is 3^i for CAKE and 2^i for LKH.

i	Create Lock MX	Solve Lock MX	Size Lock MX	LKH Enc	LKH Dec
1	280,082 µs	88 µs	41 Byte	125 µs	201 µs
2	572,785 µs	189 µs	84 Byte	188 µs	302 µs
3	822,851 µs	275 µs	124 Byte	250 µs	404 µs
4	1,130,065 µs	374 µs	165 Byte	312 µs	505 µs
5	1,377,708 µs	484 µs	206 Byte	374 µs	607 µs
6	1,539,600 µs	604 µs	247 Byte	437 µs	708 µs
7	1,909,062 µs	750 µs	288 Byte	499 µs	809 µs
8	2,231,764 µs	904 µs	328 Byte	562 µs	911 µs
9	2,544,507 µs	1,072 µs	369 Byte	624 µs	1,013 µs
10	2,751,188 µs	1,243 µs	410 Byte	686 µs	1,114 µs
11	3,134,233 µs	1,433 µs	451 Byte	749 µs	1,215 µs
12	3,387,458 µs	1,632 µs	492 Byte	811 µs	1,316 µs
13	3,705,136 µs	1,858 µs	533 Byte	874 µs	1,418 µs
14	3,974,770 µs	2,081 µs	573 Byte	935 µs	1,520 µs

Download Type (40 Bytes). For 2,187 participants, this would result in a $2,187 * 40 + 40 = 87,520$ Bytes overhead.

B. Performance on constrained hardware

RIOT OS [24] is an open source operating system that supports various hardware. Its minimal requirement of 1.5 KB main memory illustrates its lightweight nature and is one of the reasons we implemented CAKE on RIOT. Further, necessary cryptographic libraries with importance for embedded systems are available. The generation of an evaluation environment with realistic conditions is achieved by using IOT-LAB [25]. It provides a huge amount of wireless nodes with minimal capabilities regarding CPU and memory. The IOT-LAB M3 board comes with a 72 MHz CPU and 64 KB SRAM and is used for the GC and all clients. The evaluation focuses on the group management and the associated key distribution processes.

1) Memory requirements: For the evaluation we created a homogenous setup with a GC and a group of 14 clients on IOT-LAB M3 nodes. Regarding the memory requirements, the design principles of RIOT OS need to be considered, as all memory is statically reserved, including the network buffer. The required memory on the GC is defined through the number of all necessary keys within the ternary tree including the nodes. For each participant, the GC requires a total of 2,900 Bytes of data being stored, consisting of keys, IP addresses, and memory for CRT calculations and tree operations. Subsequently, the required memory for the GC is 40,600 Bytes in total, which is covered by the available memory of 64 KB in our evaluation setup. The memory requirements for participants of the group are lower. Each client requires only 2,900 Byte per connection to a GC.

2) Computational Costs for CRT: As the evaluation focuses on constrained hardware, we concentrate on time measurements for the cryptographic calculations. The most expensive operation is the IKE SA INIT message, which is caused by the necessary computation of the DH key exchange. The measurements in our evaluation setup show computation times on the IOT-LAB M3 nodes that are comparable with the times on Arduino Due in [1]. Furthermore, most actions in CAKE require only one single GSA_REKEY message carrying G-IKEv2 payloads (see [2]), which is beneficial in terms of computation time.

The most interesting new feature of CAKE is the Lock MX creation and solving on the GC and the clients in terms of computational cost. Table II shows the measurement results for the creation of the Lock MX with different tree depths as well as the time to resolve it on client side. These results are especially notable regarding a *mass entry* (see Section IV-E), where i represents the number of clients joining simultaneously. Along with the number of elements in the CRT, the computation time of new keys increases. Evidently, this is mainly caused by the higher size of the Lock MX. On the other side, the clients significantly benefit from the new method to receive keys. One simple modulo operation is needed on client side resulting in low computation time for solving the Lock MX even at its maximum size of 14 elements.

For comparison, the costs for encrypting and decrypting keys within an LKH tree are shown in Table II. It can be seen that even though the AES implementation is highly optimized, solving the Lock MX scales similar to decrypting the keys within the LKH tree. However, lowering network load with CAKE comes at the price of computational overhead for creating the Lock MX, which scales worse than LKH. Optimizing the Lock MX implementations will be part of further studies.

C. Fulfillment of requirements

Resource-constrained environments necessitate functional and non-functional requirements. The design of CAKE particularly focuses to meet these. Firstly, the ternary tree enables a reduced number of keys needed to be stored and sent. Additionally, through CAKE a re-keying is possible with one single message and the per-packet overhead is reduced. Thus, the requirements **Low Datarates** and **No 1-to-n Effect** are fulfilled. When group changes happen, the evaluation shows that CAKE requires only a minimum of messages to be sent. For group leave actions the CRT is utilized and the calculation complexity is reduced to only one modulo operation on client side. So, **Minimal Delay** and **Low calculation complexity** are achieved. The required **Minimal amount of key changes and exchanges** is realized by combining the ternary tree and the addressing scheme. This allows the minimization of required actions in case of restructuring the tree. Moreover, the **Compatibility** requirement is achieved by using the G-IKEv2 protocol, which is currently being standardized. Any client that is not capable of the newly introduced features may participate in the group by utilizing standard re-keying mechanism, while the CAKE-capable clients can still use the optimized feature set. Additionally, through the use of G-IKEv2, the **Security Properties** are met, as they are included in the standardization and therefore well studied. From this point of view, CAKE is an optimization of key calculation and transport, leaving the security parameters as they are. Lastly, the **Minimal Trust** requirement can be accomplished on a per-scenario-basis throughout the various supported authentication mechanisms of G-IKEv2.

Besides the practical applicability and its pros and cons in comparison to especially LKH, reviewing the initial design

goals and requirements shows completeness. A detailed design explanation and security assessment can be found in [4].

VI. Conclusion

In this paper, we presented the concept of an efficient group-key-management protocol that meets the requirements of resource-efficient procedures in many application scenarios like MANET. It is based on the combination of the lightweight G-IKEv2 [3] communication protocol in combination with CAKE [4] for the key exchange and key management. CAKE offers the possibility to exchange keys within a group and to react efficiently to dynamic changes of the group with low calculation effort and a low load on the network. Nevertheless, it enables confidential key distribution and compliance with backward and forward security requirements for mobile computing. The main objective to reduce the network load to a minimum is achieved at the cost of additional storage space for supplement cryptographic key material. The CRT-based key hierarchy together with a ternary keys tree structure reduce the data to be transferred especially during group leave operations. The design of CAKE delegates computational demanding cryptographic operations to the group controller, relieving to potentially less powerful group members. In today's interconnected world, this middleware technology shows advantageous in the area of secure group communication among highly constrained group members.

In the current state of research, the inefficient addressing scheme will be optimized in the next step. Apart from that, the networking overhead of the LKH extension in G-IKEv2 can be also further improved. Thus, the optimization of both in concerning scenarios is subject of future work, which will allow for a more comprehensive and technical comparison of LKH and CAKE. Beside further improvements of the CAKE prototype, the basic concept and the implementation is evaluated for building and solving the CRT System worth investigating. Finally, a more detailed analysis of the solution to post-compromise security is of great interest in the case of merging and division of groups.

Acknowledgment

We thank the MNM-Team for all the helpful comments and discussions while writing this document, especially Nils gentschen Felde. We also thank our students Edgar Goetzendorff, B.Sc., and Mehdi Yosofie, B.Sc., for assisting us during the development of the prototype and the helpful discussions.

References

[1] N. gentschen Felde, T. Guggemos, T. Heider, and D. Kranzlmüller, "Secure Group Key Distribution in Constrained Environments with IKEv2," in *Conference on Dependable and Secure Computing*. Taipei, Taiwan: IEEE, 2017.

[2] B. Weis and V. Smyslov, "Group Key Management using IKEv2," Internet Engineering Task Force, Internet-Draft draft-yeung-g-ikev2-14, 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-yeung-g-ikev2-14 (Access Date: 01 May 2018).

[3] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)," 2014, RFC7296. [Online]. Available: http://tools.ietf.org/rfc/rfc7296.txt (Access Date: 01 May 2018).

[4] P. Hillmann, M. Knüpfer, and G. Dreo Rodosek, "CAKE: Hybrides Gruppen-Schlüssel-Management Verfahren," in *10. DFN-Forum Kommunikationstechnologien*, P. Müller, B. Neumair, H. Raiser, and G. Dreo Rodosek, Eds. Bonn: Gesellschaft für Informatik e.V., 2017, pp. 31–40. [Online]. Available: https://dl.gi.de/handle/20.500.12116/476 (Access Date: 01 May 2019).

[5] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 309–329, 2003. [Online]. Available: http://doi.acm.org/10.1145/937503.937506 (Access Date: 01 May 2018).

[6] N. Sullivan, S. Turner, B. Kaduk, and K. Cohn-Gordon, "Messaging Layer Security," 2018. [Online]. Available: https://datatracker.ietf.org/wg/mls/about/ (Access Date: 01 May 2018).

[7] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification," RFC 2093 (Experimental), Internet Engineering Task Force, Tech. Rep. 2093, 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2093.txt (Access Date: 01 May 2018).

[8] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture," Internet Engineering Task Force, 1997, RFC2094. [Online]. Available: http://tools.ietf.org/rfc/rfc2094.txt (Access Date: 01 May 2018).

[9] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," 1998, RFC2408. [Online]. Available: http://tools.ietf.org/rfc/rfc2408.txt (Access Date: 01 May 2018).

[10] B. Weis, S. Rowles, and T. Hardjono, "The Group Domain of Interpretation," 2011, RFC6407. [Online]. Available: http://tools.ietf.org/rfc/rfc6407.txt (Access Date: 01 May 2018).

[11] M. Baugher, R. Canetti, L. Dondeti, and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture," April 2005, RFC4046. [Online]. Available: http://tools.ietf.org/rfc/rfc4046.txt (Access Date: 01 May 2018).

[12] Y. Challal and H. Seba, "Group key management protocols: A novel taxonomy," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 2, no. 10, pp. 3620 – 3633, 2008.

[13] S. Rafaeli, "A decentralised architecture for group key management," LANCASTER UNIVERSITY, Tech. Rep., 2000.

[14] M. Steiner, G. Tsudik, and M. Waidner, "Key Agreement in Dynamic Peer Groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 8, 2000.

[15] G.-H. Chiou and W.-T. Chen, "Secure broadcasting using the secure lock," *IEEE Transactions on Software Engineering*, vol. 15, no. 8, pp. 929–934, 1989.

[16] C. J. Antosh and B. E. Mullins, "The scalability of secure lock," in *IEEE International Performance, Computing, and Communications Conference*, vol. 27, no. 1, 2008, pp. 507–512.

[17] M. Zheng, G. Cui, M. Yang, and J. Li, "Scalable Group Key Management Protocol Based on Key Material Transmitting Tree," in *Information Security Practice and Experience*. Springer, 2007. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72163-5_23 (Access Date: 01 May 2018).

[18] G. Xu, X. Chen, and X. Du, "Chinese Remainder Theorem based DTN group key management," in *IEEE International Communication Technology (ICCT)*, vol. 14, no. 1, 2012, pp. 779–783.

[19] N. Sakamoto, "An efficient structure for lkh key tree on secure multicast communications," in *IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, vol. 15, no. 1, 2014, pp. 1–7.

[20] Z. Liu, Y. Lai, X. Ren, and S. Bu, "An Efficient LKH Tree Balancing Algorithm for Group Key Management," in *Proceedings of the International Conference on Control Engineering and Communication Technology (ICCECT)*, vol. 10, no. 2. IEEE Computer Society, 2012, pp. 1003–1005. [Online]. Available: http://dx.doi.org/10.1109/ICCECT.2012.213 (Access Date: 01 May 2018).

[21] P. Hillmann, M. Knüpfer, and G. Dreo Rodosek, "CAKE: Hybrides Gruppen-Schlüssel-Management Verfahren," in *DFN Mitteilungen Ausgabe 92*. Berlin: Deutsches Forschungsnetz, 2017, pp. 22–27. [Online]. Available: https://www.dfn.de/fileadmin/5Presse/DFNMitteilungen/DFN_92_download.pdf (Access Date: 01 May 2019).

[22] C. Matt and U. Maurer, "The one-time pad revisited," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, vol. 11, no. 1, 2013, pp. 2706–2710.

[23] Persistent Systems LLC. (2018) Mobile ad hoc networking solution delivers reliable comms, situational awareness, under rough conditions. https://www.prnewswire.com/news-releases/persistent-systems-successfully-demonstrates-flat-320-radio-mpu5-network-300634923.html/ (Access Date: 24 April 2018).

[24] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt, "Riot os: Towards an os for the internet of things," in *Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2013, pp. 79–80.

[25] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 459–464.