INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES CENTRE VAL DE LOIRE

CHEMAK
Ramy

Rapport CONFIDENTIEL ☒ NON
(cocher la case correspondante) ☐ **OUI**

Si oui est coché, signature du tuteur entreprise obligatoire

5A STI - Option 4AS

Academic Year :

2020 / 2021

A Decentralized and Reliable Election-based Key Management Protocol for Communicating Things

INTERNSHIP REPORT

Laboratory Supervisor :

LAKHLEF Hicham

Associate Professor

Referent teacher :

BERTHOME Pascal

HAUDIASYC LABORATORY UMR CNRS 7153

COMPIEGNE 60200, FRANCE

Acknowledgement

This report presents my work during this research internship. This would wouldn't have been achieved without the help of several people though.

I would first like to thank my referent teacher **Pr. Pascal BERTHOME** for his support and assistance with my application from the very start. The internship wouldn't have taken place without him.

I want also to thank my supervisors **Dr. Hicham LAKLHLEF** and **Pr. Abdelmadjid BOUABDALLAH** for offering me this opportunity. They provided me with their guidance and wise advice all along the internship.

Abstract

Despite the large deployment of IoT devices during recent years, IoT networks privacy and security remains a major concern for users. With IoT networks reaching all aspects of our daily life, IoT security came under highlight more than ever.

Researchers affiliated with the *Heudiasyc laboratory UMR CNRS 7153* developed the Multi Group Key Management protocol (MGKMP), in order to ensure message exchange security within IoT networks. This end-of-study internship is mainly a furtherance of previous research activities conducted at the host organization.

The research internship main task was to further develop the MGKMP. In particular, it aims at solving issues related to the Key Manager (KM). A second minor task concerned working on IoT security as a practiced engineer rather than a researcher. These tasks were carried out with a certain level of success, and useful results could be produced.

Keywords: Internet of Things (IoT), Key Management, energy, Cluster Head (CH), IoT security

Contents

List of Figures

List of Tables

Introduction

The use of Internet of Things (IoT) is continuously growing, and its application areas henceforth reach all societal and economical fields. IoT networks involve several applications such as smart agriculture, health care, smart homes ... etc. The number of connected IoT devices is increasing exponentially [22], and so is the amount of their exchanged data. Some of these data might be sensitive or private. To ensure their security and integrity, encryption using Group Key Management (GKM) schemes were proposed. These schemes assume the presence of a centralized Key Manager (KM), which is regarded as the central component of the system.

Previous efforts by researchers affiliated with *Heudiasyc laboratory UMR CNRS 7153* led to the development of the Multi Group Key Management protocol (MGKMP). The protocol already introduces a heap of new features and advantages. But number of its aspects are yet to be studied and improved. A part of the laboratory's work line is to to further push the research and development of MGKMP.

The internship splits into two main stages. The first stage is research oriented. Its mission is part of the laboratory's ongoing research activities on IoT security and GKM. As mentioned, it consists in the study of previous work done by the research team in charge, and elaborate upon it. The task was to study the MGKMP related issues deeper and try to adress them. Hence, the first stage was concluded with the publication of a research article in the proceedings of the *International Conference on Communications Software (SoftCOM 2021)*.

The second stage was more oriented towards parctical engineering. The main objective was to acquire a solid culture in IoT security engineering in practice, and better apprehend the challenges involved.

The content of this report is organized as follows. Chapter 1 reviews and discusses related academic works. Chapter 2 studies the architecture of

MGKMP and dig into some of its improvment axis. The main task dedicated to work the KM issue of single point of failure out is presented in Chapter 3. This includes the problematic definition (Section 3.1), solution design (Section 3.3), simulations and results evaluation (Section 3.4). Finally, Chapter 4 sums the second stage up, mainly studying different aspects of IoT security engineering.

Chapter 1

Literature Review

1.1 Related work

1.1.1 Group Key Management

Mathematical approach

GKM as a problematic has several dimensions and different types of issues to consider and tackle. Some research focused on reliable key generation issues in GKM. In [27], the authors analyzed the impact of key management techniques on connectivity and efficiency in Wireless Sensor Networks (WSNs). In order to address the issue, they propose a key generation method based on system of equations. Considering how the work was carried out to tackle poor connectivity and multiple forwarding issues, the solution is actually pretty scalable. The paper considers a system of u equations with v variables as follows:

$$\Phi^{(v)} = \begin{cases} \varphi_1(x_1, x_2, \ldots, x_v) = 0 \\ \vdots \\ \varphi_u(x_1, x_2, \ldots, x_v) = 0 \end{cases}$$

This equation has one unique solution which will be used to generate a shared secret key to be shared among the network's nodes. Solutions of equation systems actually constitute the associated keys, from which we then generate those secret keys. Two systems of equations were considered: linear equations and polynomial equations. The first system was recommended over the second one, due to its lower complexity.

Another solution based on the Kronecker product was proposed in [25]. Authors of the paper already proposed CRAPPY (Constrained RAndom Perturbation-based Pairwise keY establishment), a key establishment protocol for WSNs. However the latter doesn't consider heterogeneity in IoT

networks, neither scalability related issues. Considering the exponential growth of IoT devices and their diversity, this protocol is simply impractical. The authors utilize the matrix Kronecker product to generate private and public keys for the network's nodes. We cite the property:

math If A $\in \mathcal{R}^{\sqrt{m}\times\sqrt{m}}$ *and* $A \in \mathcal{R}^{\sqrt{n}\times\sqrt{n}}$ *are symmetric, then* $A \otimes G$ *is symmetric.*

()

Therefore, we are able to generate couples of keys for each of the network's nodes. The system assigns a couple of keys, private and public, to each node. The paper more in depth how the system exactly works. Its resilience resides in the difficulty of reconstructing the Kronecker product by any of the sensor nodes. Besides, it factorizes asymmetric encryption for messages exchange, which is rather safer than symmetric encryption.

Key management architectures

Tiloca and Dini in [23] proposed GREP, a group rekeying protocol. The protocol assumes the presence of an IDS. The latter's main function is to detect eventual attacks on the network, in order to identify compromised nodes to be evicted. However, the paper points out that nodes suspected by the IDS as compromised are considered unreliable, and therefore will also be evicted by the Group Membership Service. Thus, each node compromising or suspicion will probably induce the whole rekeying upon leaving process. Yet this topic is not covered in the paper. Even though it's beyond the scope the treated problematic, the service role in the overall problem seems under-valued. Actually, the quality of the IDS can significantly influence the global efficiency reference criteria of the rekeying protocol. On one side, a well-designed IDS is able to efficiently detect which nodes exactly are compromised. As suspected nodes are also to be evicted, the IDS can eventually raise some false alarms. Hence, optimising the IDS accuracy will decrease the number of possible false alarms, and so the number of useless rekeying process operations with all the cost that comes with. On the other side, some attack scenarios involve compromising one node at first, before propagating in the network reaching others. If the IDS is efficient enough to provide early detection of compromised nodes, it can lower the rekeying process' cost which should follow. Regarding the rekeying upon leaving process detailed in the paper, having few compromised nodes belonging to a limited number of subgroups is clearly an advantage. So we can see here that IDS role can be crucial in the key management protocol, as it can significantly contribute in the saving of the network's bandwidth and limiting the node's energy drainage.

The rekeying process, whether upon a joining node or a leaving one, is based on one central element which is the refreshing key K_R. The latter is then employed in order to generate almost all new tokens and encryption keys for group, subgroup and pairwise communications. However, the paper doesn't explain precisely how this K_R is generated, nor the measures taken to secure its generation.

Kung and Hsiao [21] proposed GroupIt, a two-tier GKM architecture. It supports multi-group networks, and embeds its own methodologies to handle device membership updates. The upper tier is resposible for key management between different groups. The lower tier is however responsible for key management within groups. The architecture considers two types of groups: device groups and user groups. Device groups incorporate devices based on their technical functionalities and security requirements. The systems also establishes user groups which contain one or multiple device groups. A device group can belong to different user groups, and every user is assigned to one and only one user group.

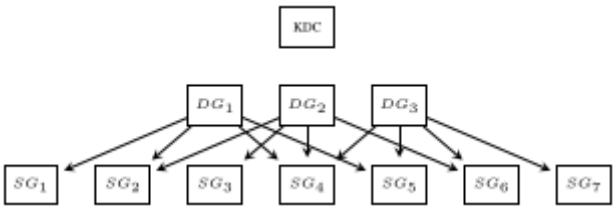


Figure 1.1: Source [21]: GroupIt initial structure overview

Every device group has its own GKM scheme, such as Logical Key Hierarchy (LKH). But above all these groups, we have a KDC which provides keys for inter-group communications. Although the proposed system considers mechanisms to ensure secure communications, forward and backward secrecy, and resistance against collusion attacks, it still has a central problem: the whole lies on the KDC. This centralization generates a single point of failure within the architecture. That's why we need to dig deeper into decentralization issues and proposed solutions.

Decentralized key management

Several solutions for decentralized Key Management were considered. Veltri et al. [26] proposed a batch-based group key management. The solution looked more at distributing the rekeying operations as events. The protocol relies on a central Key Distribution Center though, so the key management as a role remains centralized after all. Dammak et al. [12] proposed a decentralized Group Key Management architecture. This hierarchical architecture

relies on several Key Distribution Centers (SKDCs) with a central Key Distribution Center (KDC) on top. This solution already distributes the task of key management, but only partially. The SKDCs aren't completely autonomous regarding the KDC and are still dependant on it. Hence, the issue of single point of failure persists. The Blockchain technology was also considered [24, 20] to address the problem. The use of Blockchain requires from every node to keep its own copy of the ledger. This solution takes us from a point where a single entity is handling the key management, to a situation where practically all network nodes are acting as Key Manager. Giving the need to add a ledger block on every rekeying operation, this introduces a significant processing overhead (due to required Proof-of-Works) and networking overhead (due to required ledger update).

A cluster based Key Management solution was suggested by Khan and Anandharaj in [13]. It uses a cognitive key management technique to increase energy efficiency and scalability. The work considers Cluster Head (CH) election process and highlights the need for cluster maintenance. However, the proposed election process adopts a classical network routing approach. Besides, it does not consider security risks related to this cluster based scheme, nor its mitigation. The algorithm's analysis is focused on energy efficiency, with few regard to security.

1.1.2 Cluster Head selection

Actually, Cluster Head based architectures have been long studied. Heinzelman et al. [14] considered network clustering as a network routing solution. They proposed LEACH (Low-Energy Adaptive Clustering Hierarchy) to address the energy dissipation issue in node-to-base station communications. It suggests a cluster-divided network with one node in every cluster acting as local Base Station (BS), dubbed CH. Different CHs collect data from their fellow cluster members, and transmit them to the main BS. Extensions of this scheme were then proposed [9, 18]. Other algorithms for CH selection seeking to optimise energy consumption and load-balancing were suggested in [11, 15]. These algorithms utilize node's energy capabilities to balance probabilities for CH selection among other cluster nodes. But all these studies aim to optimise energy performances with no regard to security. They were basically developed to save radio transmission overhead, thereby they deal with network and data routing oriented problems. However, our interest for this scheme is foremost driven by our need to solve the Key Management problem for group communication and provide a decentralized model for it. Moreover, the CH election in previous works is based on energy, communication and networking related criteria. Besides, they mainly consider homogeneous environments in which CH are to communicate with a fixed Base Station (BS) for application reasons. Hence, the described models are

not necessarily fit for use as a security oriented scheme (cryptographic key management in our case) for heterogeneous IoT networks. In our use case, we tend more to think over the architecture from a security point of view.

1.2 Identification of remaining challenges

In literature review, we considered both, decentralized key management as a challenge and network clustering as a scheme. Based upon this, we can notice that these works are mostly energy performance oriented. We need key management and cryptographic keys to ensure security in the first place. Despite that the main problematic, key management, is actually a security problematic, relevant researches are usually conducted with a networking and energy mindset, sometimes discarding security concerns [29, 28, 19]. This work aims at bringing back the security perspective to the front view, besides the energy aspect. Thus to propose a solution which offers a satisfactory security-efficiency compromise.

Chapter 2

Multi Group Key Management Protocol

The main mission of the internship is to continue the underway research work on the Multi Group Key Management Protocol (MGKMP), being conducted within the host lab. Section 2.1 presents key elements the work done so far. It's kind of a literature review specific to the lab's scientific production. The rest of this chapter sums up the first assignment, which consists in the in-depth study of the protocol and identification of work yet to be done.

2.1 Overview of MGKMP

The MGKMP is organized in three layers. The first layer is intended for node and key management. Each node is assigned to a subgroup S_j^i of the group G_i. Managed keys can be categorized into Data Encryption Keys (DEKs) and Key Encryption Keys (KEKs). These keys are either used for device-to-device communications (pairwise keys) or group and subgroup communications. The second layer is intended for subgroup management. This partitioning considerably decreases the nodes' storage overhead. The third and last layer is intended for service and group management. Groups are chosen according to a unique combination of services. Fig. 2.1 illustrates the MGKMP architecture. Still to note that subgroups layer is logical and totally transparent to the application layer.

The protocol aims at ensuring five different properties: scalability, efficiency, heterogeneity, collusion resistance and forward and backward secrecy. The two last properties are security oriented, whereas the first three ones are used to evaluate the protocol's performance. Forward secrecy is guaranteed when a leaving member of a group has no longer access to his former group communications. Backward secrecy is guaranteed when a joining member of a group has no access to his group's old communications. A collusion

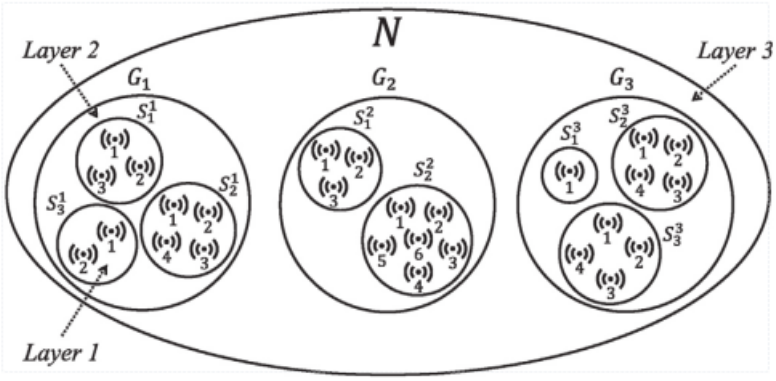


Figure 2.1: Source [17]: Example of a network partitioning

attack unfolds when multiple compromised nodes (evited or not) cooperate using their individual pieces of information to gain illegitimate access to the group key. To curb these security risks, the protocol relies on keys revokes and rekeying operations. Every time a node is joining, leaving or evicted from a group, all pairwise keys associated with these node as well as related group keys are revoked and new ones are generated. Rekeying operations and group memberships are handled by the Key Manager (KM). This makes the KM the protocol's security backbone.

Among key contributions of MGKMP is its compatibility with heterogeneous networks. Giving the increasingly sophisticated and inter-connected nature of IoT networks, this is a major asset.

2.2 Optimization of MGKMP

These working tracks consider mainly the optimzsation of the MGKMP protocol. The idea consists in redesigning some aspects of the protocols features, and then comparing the new design's complexity and other properties with the original design. Presumably, each optimization brought should be compared apart, and eventually, a combination of those could be tested. The ultimate objective is to produce a better optimized version of MGKMP in terms of security, resilience, flexibility, efficiency and scalability.

2.2.1 Analysis of an n-tier architecture

The first subgroup layer in the paper aims at grouping devices by physical capacity and power. Thereby, it decreases energy drainage for devices. But this extra layer piles up complexity to the overall architecture and cryptographic key management, whether those used for group or device-to-device communications. This complexity increase shadows on the raising number

of required messages during rekeying operations, and therefore, an increase in network traffic and bandwidth consumption. However, network traffic indulged on a network adapter is itself a factor of energy drainage for the device.

Now let's imagine an extension of the suggested protocol to a 4-tier architecture. This might contribute to the enhancement of the storage overhead and computation effort for a device node. But it would probably lead to ever further utilization of network bandwidth as well, and so, the device battery might waste on network adapters what it had saved from its CPU. However, the study of both aspects of the protocol's performance shall lead to a proper compromise.

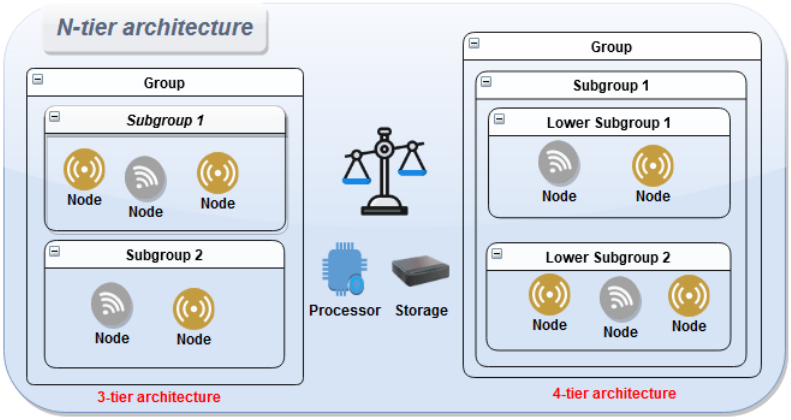


Figure 2.2: 3-tier architecture vs. 4-tier architecture

A naive question to sum up the issue would be:

> What consumes more energy: network adapters or CPUs ?

()

It is agreed upon though that network adapters consume more energy than processors. Since we are considering heterogeneous IoT networks, the devices we are facing can be very much different. Therefore, it's going to be difficult to say which physical component is more likely subject for optimisation and energy saving enhancement. Therefore, this task will most likely requires an overview and analysis of common devices' physical capacities in order to compare with the analysis of bandwidth and CPU utilization influence on energy drainage. The study is not to be necessarily restricted for the computation and network capacity. Other elements might interfere in the carried out process, and so they are to be taken into account as well.

Similarly to the Storage Capability Evaluation Function, we can define for instance a look-alike function for the Network Capability Evaluation Function. Hence, the ultimate Capacity, according which nodes are attributed to subgroups, will be the average of these two functions. The average does not have to be static though. It can very well be dynamic according to the device's physical characteristics, and common patterns can be defined for this purpose. In case of a successful theoretical analysis, implementing an experimental prototype for a real-life use case shall make perfect sense.

The work on this axe of optimisation can contribute significantly to the scalability and efficiency of of the proposed solution.

2.2.2 Re-order algorithm upon leave

In the aftermath of a node's exit out of the network, a rekying process upon leave in launched by the Key Manager. In case a subgroup's cardinal decreases down to a certain threshold, then it's merged with an other subgroup. Concretely, the solution looks for an subgroup to merge with, creates a new subgroup to which it assigns the nodes from the two merging subgroups, and finally removes the latter two subgroups.

Sometimes this merge process comes after the eviction of a compromised node, in which case the two merging subgroups could be compromised. In this situation, and for security reasons, it's rather pertinent to have them both removed and create a new one with freshly generated keys. Nevertheless, this is not always the case. When no security breach requires it, this would only be a waste of computational power and bandwidth and loss of efficiency.

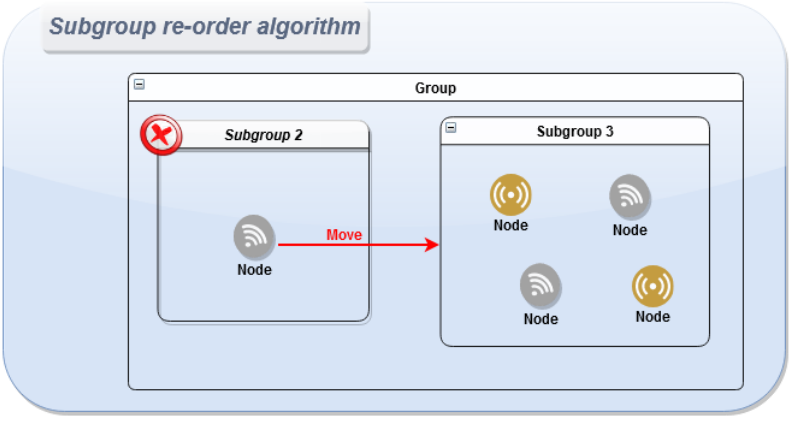


Figure 2.3: MGKMP subgroup's re-order algorithm

A simple example to illustrate the kind of optimisation to bring on, is when

a subgroup S_1 of cardinal $n = 2$ is to merge with another subgroup S_2 of cardinal $n = 8$, assuming we have a threshold $thr = 2$. In this case, it would rather much more efficient to just assign the nodes from S_1 to S_2 and provide them with the necessary keys, instead of creating a new subgroup S_3. In this case, the 8 initial nodes from S_2 are saving all the computational power required for key generation as well as bandwidth for message exchanges for joining the of S_3. These 8 nodes have practically nothing to do. Moreover, the KM would have saved the computational energy for the creation of S_3, and its key generation.

2.2.3 Sub-grouping sequences

As mentioned previously, layer 2 is intended for subgroup management. Nodes within a group are subdivided into multiple subgroups based on their minimal capabilities *mc*.This *mc* value is determined through two main algorithms, referred as sequences: powers of two and Fibonacci.

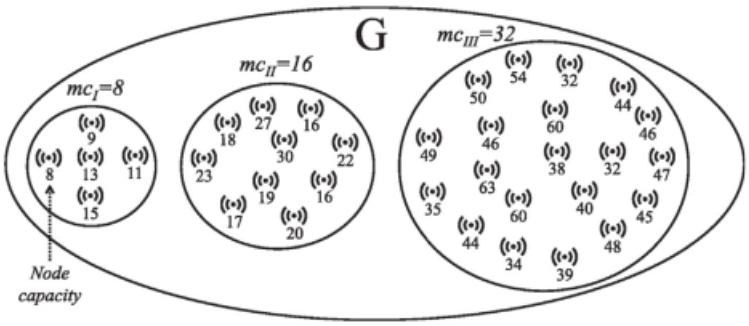


Figure 2.4: Source [17]: Example of a group partitioned using powers of 2 sequence

This subgrouping sequences could be studied further, in order to eventually develop new efficient sequencing algorithms.

2.3 Internal liabilities of MGKMP

The corresponding working tracks seek to enhance the protocol's security by looking into its performance when applied into practice. Many features of the protocol are demonstrated in a abstract and theoretical way. But some of the slightest behaviours in their concrete implementations might turn out to be extremely costy. Some of these features interact with each other, which can have implications on how they should be defined for practical implementation. The hereby considered problematic also deals with efficiency enhancement as well as security enforcement. So the point is to look out how practical the theoretical definitions of MGKMP are, and how to adjust

eventual inconveniences.

2.3.1 Refresh key generation

> The generation of random numbers is too important to be left to chance.
>
> (*Robert R. Coveyou*)

One of MGKMP's key properties is forward and backward. The main mechanism implemented to ensure it is the group rekeying process. As described in Section 2.1, the new key is generated via a KDF which takes the old key and a random key K_R as parameters. From a cryprographic point of view, the random generation is crucial to the whole process. Any vulnerability can blow the forward and backward secrecy up. An evicted or leaving node already has the old key. So if it gets hold of K_R, it can easily compute the new key. A recently joined node already has the new key. So if it gets hold of K_R, it will be possible to deduce the old one.

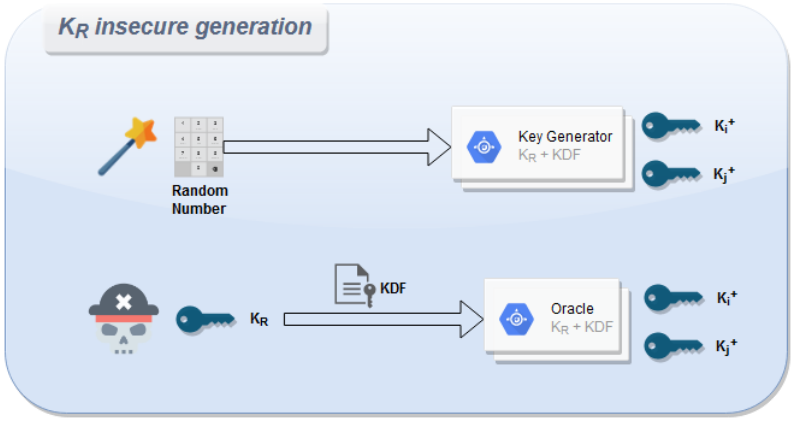


Figure 2.5: Insecure K_R generation

Fig. 2.5 illustrates how an attacker who is able to guess K_R and knows the KDF used, is able to compute newly generated secret keys by using some cryptographic oracle.

2.3.2 Node's join authorization & pre-secure channel

The joining of a new node comes with two assumptions. First, the joining node is reliable and trustworthy. And the second implies that we dispose of a pre-existing secure channel between the Key Manager and and the joining node. The first assumption don't take into consideration the possibility that

a node can already be compromised before it joining the network. Therefore, it burdens the risk of having all the new node's keys systematically compromised. The second assumption assumes the communication's security between the Key Manager before having the node's join process through. The non-compromising of all encryption keys related to the new node depends on this channel's resilience. In both cases above, we have a systematic key compromising situation just in the aftermath of a node's join operation. This will eventually lead to have this node evicted afterwards, and hence, launching a rekeying upon leaving process.

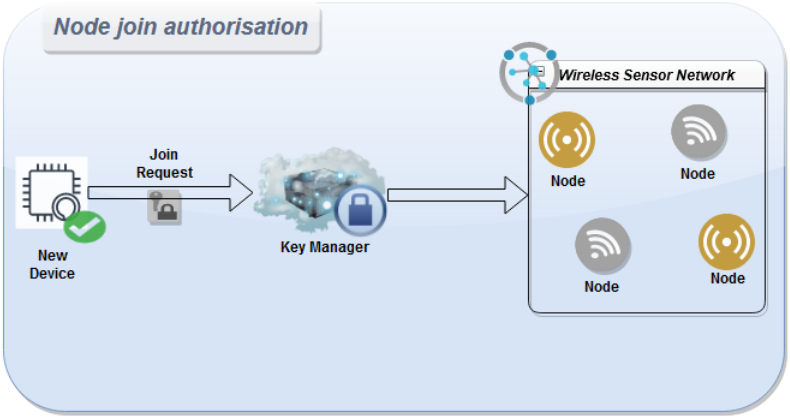


Figure 2.6: Node's join authorization

2.3.3 Inadequate choice for cryptographic algorithms

Depending on the data to hash or encrypt, algorithms have to be considered with caution. It's easy to mistake cryptographic algorithms for an independant and irrelevant problem in our case. But MGKMP is protocol which directly deals with cryptography. When we look at common cryptographic issues, they are more related to the system implementation, rather than the algorithm itself. Although the algorithm's choice remains beyond the scope of this research work, it has to be thought in advance to facilitate the enginneer's work, who is actually implementing MGKMP. The point of this section is to underline the influence of protocol's design in hardening or easing the resilience of implemented cryptographic algorithms. It doesn't assume any lack or flaw in the protocol, such conclusion can only be the outcome of a dedicated study.

Insecure key exchange

Key exchange protocols solve a crucial problem, which is sharing a secret key between two communicating entities over an insecure channel. The

most widely used protocol today is Diffie-Hellman. It can be implemented using discrete logarithms and elliptic curves. Even though the protocol is mathematically resilient, it's absolutely helpless against Man-in-the-Middle MITM attacks.

Insecure hash functions

Hash functions are very used in the MGKMP, but they have to be consiered carefully. Hash algorithms like MD5 or SHA-1 are utterly broken. SHA-2 hash family is more performant and safer than SHA-1, for its collision resistance. But once again, depending on the context, attacks like reduced-round collisions [**?**] or length extension attacks aren't excluded.

Message authentication codes

In relation with Section 2.3.2, the use of Message Authentication Codes (MAC) is almost unavoidable. But that's a cryptographic challenge itself [**?**] and requires robust design of the related feature in MGKMP. For instance, prefix-MAC constructions is totally insecure even with SHA-2. Only SHA-3 algorithms allow the implementation of safe MAC systems.

Signature algorithms

The most robust signature algorithms available actually rely on public key cryptography. Attacks on RSA such as common modulus, hasted attack, Wiener's attack or common factor attack are leathal. How trustworthy asymmetric cryptography can be, it still has its own weaknesses. What is interesting is that these flaws aren't related to the crypto-system itself, but to its implementation. This means that these vulnerabilities should be carefully considered when designing a security system based on cryptography, as it's the case here.

Lightweight cryptography algorithms

Once more, the system designed here directly concerns cryptography. The resource-constraint nature of the IoT rather drives the attention towards lightweight cryptography algorithms [**?**, **?**]. The choice of these algorithms has to take into account context requirements, since their characteristics differ according to the use case [**?**]. But something recurrent in these lightweight algorithms, is the curb of their resilience in favor of performance gain. This can be for example round reduction in AES algorithms, or the use of easy-to-generate initialization vectors (IV) in stream ciphers. If the host system (the MGKMP architecture in our case) allows it, such measures can lead to related-key attacks on AES [**?**, **?**] or IV attacks on CBC and ECB.

2.4 Analysis of the protocol's ecosystem

> The first key to any effective security game plan in knowing what you're up against
>
> (*Pete Herzog — The Open Source Cybersecurity Playbook*)

So far, the protocol's related research work focused mainly on the conception the protocol's processes and the exclusive definition of its internal core functionalities. In many cases, the development undertook a mathematical approach to identify different problems and solve them. However, the protocol shall be integrated in a probably complex environment, where its own functioning will influence and be influenced by external actors and components. This protocol already solve several critical security issues and significantly decrease the probability of an IoT network's compromise. However, it's only an actor in a bigger interconnected and hostile ecosystem.

This section seeks to provide some ideas for the analysis of these interactions, measurement of their impact on the well-functioning of the protocol, and suggestions to improve the protocol considering the conducted risk analysis. The point is to see the problematic from different angles in order to better apprehend the protocol's ecosystem and the stakes which come along, in order to have a system practically compatible with any or at least most common environments seen in major use cases. Furthermore, some aspects of the protocol could be possibly enhanced in order to cover up for attack vectors coming from those actors and reduce the attack surface.

The previous threats and vulnerabilities enumeration demonstrate some of the possible attack vectors for cryptographic key compromising. Since our protocol aims mainly at securing cryptographic key exchange and encrypting data transmission, then these attack scenarios could of interest for study. In these examples, we demonstrate that an attacker does not necessarily have to break the protocol's design itself in order to compromise the managed cryptographic keys. But he can however look for actors in the overall ecosystem interacting with our protocol, and take advantage of its eventual weaknesses.

The protocol's environment to consider is pretty large and can include a variety of inter-connected actors interacting with our nodes and the Key Manager. Tiloca and Dini [23] already described an architecture in which the KM performs its duties. This architecture involves a Key Management System (KMS) operating above the KM. On the same level as the KMS, we have a Group Membership Service (GMS) keeping track of which nodes belong to the group. In our architecture, we can very intuitively imagine that this same service would also be keeping track of which nodes membership regarding subgroups as well. When a new node joins in or is evicted out

of the group, the GMS should have a process to carry out in order update its membership lists. Still on the same level as the KMS, the architecture includes an Intrusion Detection Service (IDS), assuming the task of identifying malicious nodes and report them the GMS to have them evicted. Finally, on top of the three services (GMS, KMS and IDS), we have a group controller (GC), which role is to oversee the well functioning of these services. Fig. 2.7 illustrates the architecture described by the authors of the GREP protocol.

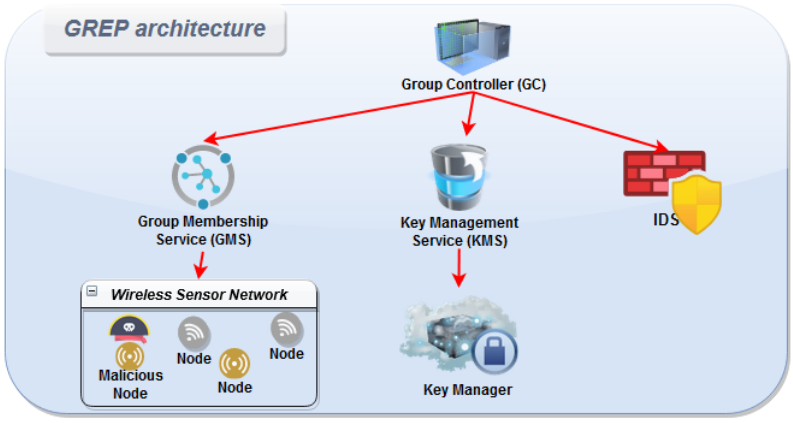


Figure 2.7: GREP architecture

However, a realistic infrastructure in which the Key Management protocol shall be integrated is definitely much more sophisticated that the one described above. Fig. 2.8 illustrates, in a non-exhaustive way, the diversity and complex interconnection of a realistic environment. We can see below the Key Manager and the Wireless Sensor Network which are embedded in the core of the protocol. But all around we have a dynamic and active ecosystem of several entities interacting with both. Among those entities, we have a firewall, through which go all transmissions from and to the group. There is as well the GMS and an IDS, as already discussed. In order to mitigate risks of a malicious node joining the group, we can imagine an Identity Access Management system in order to check out a node's join request legitimacy. This IAM service can also assume other responsibilities in our environment. The IAM service, depending on the use case, can be substituted with an Access Control Server (ACS). As it's more and more common, we can assume that our network is somehow related to some Cloud-based service. Therefore, additional security constraints are also to be considered by the protocol. Furthermore, the node's architecture itself is an element worth attention, whether form a software or a hardware point of view. Further features such as logs generation by the KM for eventual forensics analysis can be considered too.

From previous analysis, a number of attacks such as the low-level Sybil

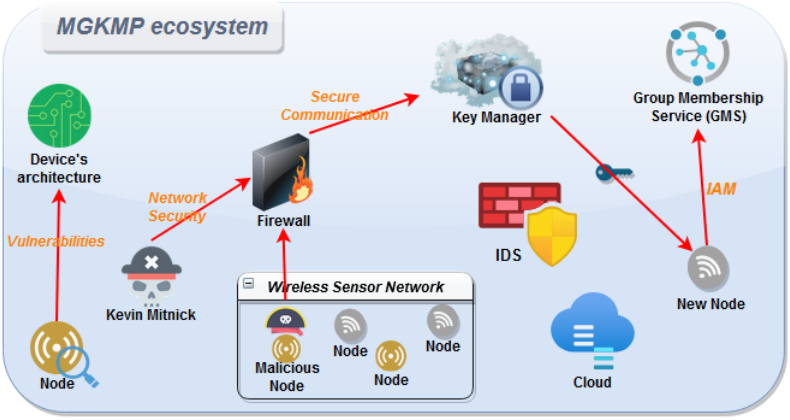


Figure 2.8: Key Management protocol's ecosystem

attack and RPL routing attacks are actually conducted by malicious nodes within the network. Our protocol doesn't interfere in the detection process of veiled nodes. But it does make sure, throughout the rekeying upon leave process, that this node is indeed evicted and cannot access the network's communications afterwards, ie. ensures forward secrecy.

Firewalls are a central element in IoT networks in order to filter out unnecessary or malicious network frames originally destined for some node, especially with the ever coming-back energy constraints. Yet, firewalls are also an attractive concentration point of network traffic. The compromise of the firewall, even partially through a foothold, might to the compromise of other nodes, if not the whole network. That's why communications going through in and out should all be enough secured and authenticated, at risk of jeopardizing internal communications on the group. Among other reasons, this can significantly mitigate risks of insecure neighbor attacks. The device-to-device communications requires a prior steps of router discovery and address resolution. A firewall in the inside can make sure that these messages aren't detoured to a malicious destination. The firewall can also be deployed to monitor these internal communications themselves, in which case node-to-node and KM-to-node messages will all go through the firewall. If the network's security is not robust enough, a hacker would be able to eavesdrop on some communications allowing more damaging attacks such as fragmentation replay and duplication attack or just tampering with frames. The latter is actually a very serious threat as it opens the door for a bit-flipping attack.

The protocol is essentially designed to ensure cryptographic keys management and exchange between different nodes and the KM. As briefly explained above, the network's infrastructure can sometimes be easily eavesdropped on. This ease in intercepting transmissions might very well allow

an attacker to measure and analyse time required for nodes to perform protocol related encryption operation, exposing the protocol to a category of side-channel attacks, known as Timing attacks. This also intercross with the previously part mentioning the Timing attack.

In Section 2.3.2, we discussed the problematic of the pre-established secure channel between the KM and new joining node. A reliable communication over this channel requires an end-to-end security on the transport layer. The purpose is to make ensure reception of a message by the desired destination while authenticating the sender. Here we go back to Message Authentication Codes (MAC) and message signature algorithms, already discussed in Section 2.3.3. This measure concerns joining nodes as well as device-to-device communications. Device-to-device communications should also bare the network security aspect since they are subject to session establishment and resumption attacks. These attacks can badly affect the integrity of, among others, some Data Encryption Keys (DEKs).

To quickly sum up the previous, we have a recurring problem, which is node compromising. It's here where the importance of malicious node detection stands out. For this point, we already assumed the likely presence of an IDS in our ecosystem. We have also considered a firewall filtering internal communications of the group, in which case it can also monitor the network traffic in the same time to detect abnormal behaviours. An independent monitoring solution, such as a Zabbix server, could be considered too, while its output can be used for veiled activities detection. Depending on which solution is to be used, we need to make sure that different operations carried out by the Key Management will always be compatible with its environment. We can even consider different versions of the protocol for each use case, with eventual optimisations for each facilitating the bad nodes hunt. All of this requires a standalone study and analysis to select what can possibly be done on this path.

Besides, all previously mentioned entities such as IAM service, firewalls and ACS are actively interacting with the group nodes and the KM, as already mentioned. To some extent, and depending on the situation, some on them could even be considered as insider actors of the group communication. Hence, they are technically members of the group, requiring encryption keys provided by the KM on their turn. As a consequence, an other study and analysis could be carried out to precise (i) which keys exactly these elements will need to efficiently assume their functions, (ii) how are they going to be distributed by the KM and also regarding the other nodes, and foremost (iii) how the KM should handle them and manage their keys. This study shall improve the implementability of the protocol and make it easier to integrate in a concrete industrial application.

Finally, the Key Management protocol already takes into consideration dif-

ferent characteristics of IoT devices, such as their low computation power or constrained energy capacity. Those considered characteristics are mainly physical though. There is much more in the devices to be studied in order to analyse its implication of well functioning of the protocol. In case of popping up issues, optimisations and improvement shall be made on the protocol's design to make it further operational. This study turns around the devices systems, from the hardware as well as the software point of view.

On the software level, we have already mentioned some typical threats including malwares. A malware-infected node can easily have all its stored encryption keys compromised. To this purpose, we can think of some security mechanisms regarding how keys are to be stored on the device. These mechanisms have already been thought in the protocol, but only regarding the storage overhead. So a further study of this part can very well enhance the integrity of the protocol. From the hardware point of view, two main elements should be considered to assess the protocol's efficiency and usability in its environment: (i) the device's firmware and (ii) the processor's architecture. The hardware study is crucial as it allows the assessment of the risk level regarding a possible rootkit compromising. Same as for malwares, a rootkit infected node can easily have all of its encryption keys compromised. Nevertheless, rootkits are far more lethal than high level malwares and foremost, they are extremely stealthy and hard to detect, if they are detectable. Rootkits raise the interest of studying the processor's architecture besides the firmware. The interest in considering processors architectures could come from other threats too, like the Timing attack. An analysis can conclude on whether the protocol's key exchange and storage processes take this kind of threat into account or not, then conclude on which enhancements, if possible, can be added to make the protocol more resilient.

Chapter 3

Key Manager: Single Point of Failure

In Chapter 2, we described a number of issues related to the MGKMP, without going deeper or proposing solutions. One more issue concerns the Key Managar (KM). For the latter, we decided we would be working more in depth on it, and propose a solution to solve the problem. This Chapter represent the internship's main mission, which led to the article publication.

3.1 Problem description

The whole architecture of the proposed solution above lays on one central component: the Key Manager. This vital component is to handle on its own the complete key management infrastructure and all the processes defined in the protocol, such as the rekeying process or key generation. Since we are considering a multi-group scheme, each group is handled by a Key Manager.

The Key Manager has to permanently satisfy the three main security properties: Confidentiality, Integrity and Availability (CIA). Any failure or the compromising of this machine will systematically jeopardise the whole security infrastructure of the group. A security breach of the KM can have devastating consequences, leading to the leak of the stored encryption keys used by respective group fellows.

A typical scenario is when the KM is down. This could be due to some malfunctioning in the system, or even a cyber-attack like a DDoS attack, or just result from some overload submitted to the KM, since it coordinates all nodes in the network. It could also occur upon an incident like a blaze in a datacenter [10]. In this situation, no rekeying process is going to be possible, as this operation is on the KM's responsibility. Therefore if a compromised node is detected, either by neighbouring nodes or by a third party defense system like an IDS, it won't be possible to have it evicted from the group. It wouldn't be possible to have the group tided up neither. Basically, what-

ever device has detected the compromising, it would automatically report it to the KM in order to have it settled. However, since the KM is down, this eviction request will never be treated, and the cleaning up operation will never be carried out. Moreover, as long as the KM remains down, the compromised node will remain a member of the group having all group's communication it's allowed to compromised. Furthermore, the node not being evicted once it's uncovered, it can spread into the network compromising all group related keys it hasn't yet in its inventory.

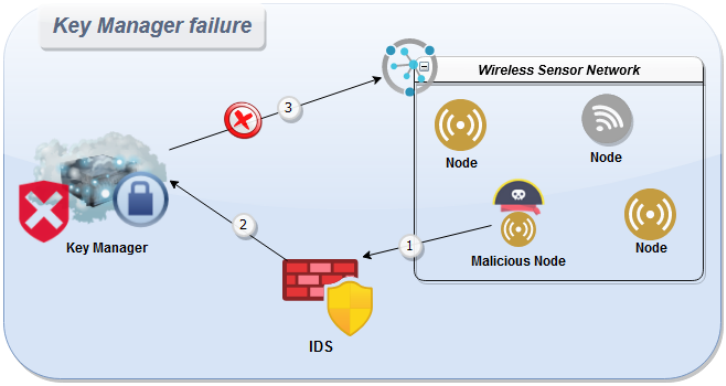


Figure 3.1: Key Manager failure scenario

One would want to say that if we add some redundancy to the KM, that would solve the problem. If the main KM is down, then we have some backup clones to take over until the main one is on track again. Nevertheless, this isn't necessarily guaranteed, because availability isn't the only security requirement to satisfy as mentioned above. To illustrate this, we can imagine an interference in the communications from and to the KM. A malicious node or a third party attacker can spoof the KM's identity and hence, conduct a Man in the Middle (MITM) attack. If the KM is compromised, then so are all operations it is carrying out and the infrastructure as a whole would be rotten to the core. Even if the communications are encrypted, the attacker can still tamper with a rekeying process upon a node's leave. Actually, if the attacker has access to the leaving node's keys, then he can forge rekying messages with the old revoked keys. That way it would be as the whole rekeying operation hasn't took place, and the leaving node will have a theoretically unauthorised access to so many communications, making them all compromised. An attacker acting like a proxy for the KM can also intercept messages for a joining node. But as a pre-secure channel should be established between the KM and the new node, the way he can compromise these messages depends on how this channel is implemented.

This attack is a bit more difficult than the previous to mount though, since the attacker has to manipulate data (eg. the ARP table) for every node

and other entity that might be in interaction with the KM. Unless the KM is placed behind a single component, such as a Firewall, then the attack difficulty would depend on the resilience of the latter.

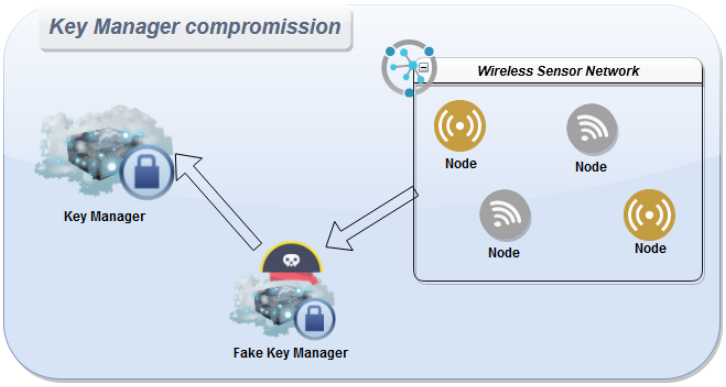


Figure 3.2: Key Manager compromising scenario

To sum up, we have two main problems concerning this architecture making out of the KM a single point of failure. The first one concerns the load imposed over the KM. It has to coordinate every node from groups and subgroups. Every join or leave operation is validated by the KM, and the consequent rekeying operations are also handled by it. In the original design, we already assume that the KM is powerful and robust. But such performance assumptions can be costy, and still, it remains a system with its own physical limits like any other system. All this makes the KM a ***single point of charge***. Second, we also make the assumption that the KM is trustworthy. The fact that all network membership evolution have to be validated by the KM, and the rekeying process is entirely handled by it as well, gives it a large base of trust. The MITM attack described above shows how critical this trust can be. This makes the KM a ***single point of trust***. A way to mitigate these risks, is to design a decentralised architecture with a distributed base of trust and a distributed load of charge.

3.2 Blockchain-based scheme

Kandi et al. [16] suggested a solution based on the Blockchain technology as explained previously (see Chapter 1). This solution however might end up too costy for IoT networks. One one side, the Blockchain implies that all nodes, regardless of their storage capacity, will have to store the ledger, which will definitely cause a significant additional storage overhead. The more traffic we have (nodes joining and leaving), the more blocks have to be added. This makes the storage overhead, in addition to its height, dynamic and of an unpredictable growth. Moreover, these transactions will

introduce a consequent network traffic. Hence, an increase in bandwidth consumption and drainage of node's already-constrained resources. Besides the performance hindering, a hacker can see in this an opportunity to conduct a parallel DoS attack. By maliciously interfering with the network traffic, one can instigate false join and leave requests, which will be translated into appended blocks to the ledger. Abusing this kind of interference might consume the total node's storage capacity. On the other side, each block append to the ledger requires a computational effort, a Proof of Work. PoW operations or even PoS operations in the Blockchain aren't the most trivial to compute. They require a certain minimum threshold of computational power, and the way the Blockchain technology was designed in the first place, tend to have such threshold as high as possible, which is exactly what we want to avoid. Even with some lightweight versions of the Blockchain, it doesn't resolve entirely the problem and it may very well add further problems. Those lightweight versions are actually based on constraint-reduced PoW or PoS. However, this can open the door for some security risks since transactions would be easier to validate, and so the vote process will be more accessible. Hence, malicious nodes will be more likely to conduct veiled activities, and we're not even sure if the computational capacity problem is solved.

3.3 Election-based scheme

Another strategy for distributed charge and trust for the Key Manager, is to have an election-based scheme, where any node can be the KM. This way, we mitigate the risk of single point of failure present in the original design on one hand. And on the other hand, we don't necessarily have all nodes assuming the costy role of a KM like in a Blockchain-based scheme. The main idea is that one node among the group will assume the responsibility of a KM (which is already possible in the centralised version). This node has been chosen and approved by the others. This choice is based on some node-related technical criteria, which make it eligible for this role. In case the KM down or compromised, any other eligible node can take over. A KM can also choose, if the situation allows and requires it, to hand over its responsibilities in case it is or will be suffocating. The KM still uniquely assumes the responsibility of the key generation process during a rekeying operation, but it can share the burden of newly generated keys distribution with the others. In this scheme, key generation has also been improved and optimised for less energy drainage. In addition to the role of a KM, we also define a deputy, which is the privileged node to take over if the KM is, for some reason, deemed to step down. This deputy has also a monitoring role over the KM.

In Section 1.1.2, we discussed several Cluster Head (CH) selection algo-

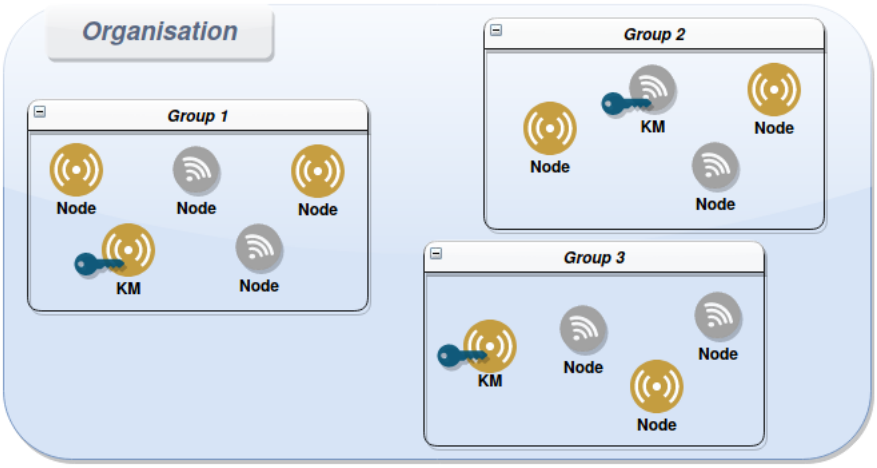


Figure 3.3: Election-based scheme architecture

rithms. However, these algorithms were mostly designed to address network routing and data exchange problems, such as LEACH (Low-Energy Adaptive Clustering Hierarchy) [14] and related extensions [9, 18]. Other algorithms for CH selection seeking to optimise energy load-balancing were proposed in [11, 15]. These algorithms utilizes node's energy capabilities to balance probabilities for CH selection among other cluster nodes. But all these studies aim to optimise energy performances with no regard to security. They were basically developed to save radio transmission overhead, thereby they deal with network and data routing oriented problems. However, our interest for this scheme is foremost driven by our need to solve the Key Management problem for group communication and provide a decentralised model for it. Moreover, the CH election in previous works is based on energy, communication and networking related criteria. Besides, they mainly consider homogeneous environments in which CH are to communicate with a fixed Base Station (BS) for application reasons. Hence, the described models are not necessarily adapted for use as a security oriented scheme (cryptographic key management in our case) for heterogeneous IoT networks. In our use case, we tend more to think over the architecture from a security point of view. The main security criterion the CH must satisfy is availability. We are not using the CH a as relay between nodes and some BS. We are rather building a security infrastructure to manage cryptographic keys in order to secure both group and device-to-device communications between different nodes. When rethinking the election-based scheme from this prospective, we actually end up with many additional constraints than those already considered.

In this section, we propose a new complete election-based scheme adapted for Key Management in multi-group IoT networks. The point is to success-

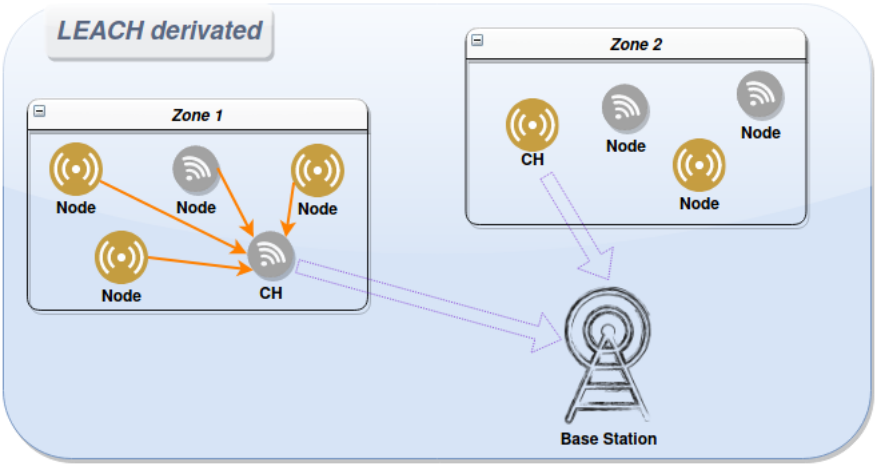


Figure 3.4: Overview of LEACH-based architecture

fully define a decentralised election-based infrastructure for Key Management which (i) solves the problem of KM single point of failure, (ii) satisfies the heterogeneous and energy-constraint nature of IoT networks, (iii) considers the main objectives we target from a security point of view, and (iv) tries to inherit as possible the features offered by previous work (even thought not all of those are needed for our use case). In the very first basic approach, the idea was to have one unique component that handles the Key Management, where came the problem of single point of failure from. In the first attempt to solve the problem, the Blockchain technology was considered, where the energy consumption problem came from. So we basically went from having one single node doing this heavy important task in the first case, to a state where all nodes are assuming this role regardless of their capabilities in the second case. The approach we would like to propose tries to combine both previously mentioned approaches in order to suggest something in between. The advantage of the election-based approach is that it have one particularly capable node acting as KM (the first approach likewise) while actually allowing any node in the network to take part (the second approach likewise). The fact that this node will be elected and validated guarantees the reliability of the KM from both security and performance point of view. Therefore, the architecture should support decentralised decision making, and thus enhance failure management. As for the KM election, it should take into consideration security concerns through consensus-based trust. Finally, to maintain scalability and IoT performance requirements, the architecture must be flexible and involve minimum vote processes and communication overhead.

3.3.1 Technical eligibility criteria

Basically, any node can be the KM and that's the basis to have a distributed system. However, those can be heavy and expensive responsibilities to assume, which require minimum capacities in networking, storage and processing. Otherwise the node could be merely qualified as single point of failure by default. To ensure that a candidate node will be up to it when qualifying for KM, we define two new functions: Processing Capacity Evaluation Function (PCEF) and Networking Capacity Evaluation Function (NCEF). Those two new functions will be aggregated with the already defined Storage Capacity Evaluation Function (SCEF) as an input for the Capacity Evaluation Function (CEF). The latter's output is what gives us the bigger picture about the node's real technical capabilities and qualification for the KM's assignments. The output of the CEF is a score, which has to be above a minimum threshold for a node to be eligible for the role of the KM. The final score value shall be correlated with the node's energetic capacities as well. This settled, we ensure that an elected node for KM is indeed technically and energetically reliable.

The CEF score should be an average value totally blinding regarding the input values. For this purpose, the input values could, eventually, be aggregated with a salt to add some noise. There are two main reasons why the CEF's output is a single dark value. The first reason is saving network bandwidth and energy. In this form, a node will be just broadcasting a number, instead of a long message detailing its capacities. Besides, this increases accuracy when providing a reference scale for decision making (voting). The second one is security related. One is never sure whether a pre-compromised node is present in the group like a Trojan horse, or an outsider attacker is somehow eavesdropping on the group communications. Therefore, it's more secure if the CEF score doesn't leak any data about a node's real capabilities in networking, storage and processing. The acquisition of such information can be used to mount other specific attacks, and so very harmful. A typical example, is side channel attacks in cryptanalysis. Knowledge of some physical properties of a device can significantly increase the efficiency of these category of attacks, in a context where cryptographic choices are critical (see Section 3.4).

Technical description of CEF score

The CEF score is computed based upon three inputs SCEF, PCEF and NCEF correlated with an energy score.

First of all, we start with the SCEF since it's already at our disposal. The KM, as its name suggests, will have to manage keys. Among other things, it stores all cryptographic keys used to secure communications within the group, as each group has an independent KM, and exchanges with other

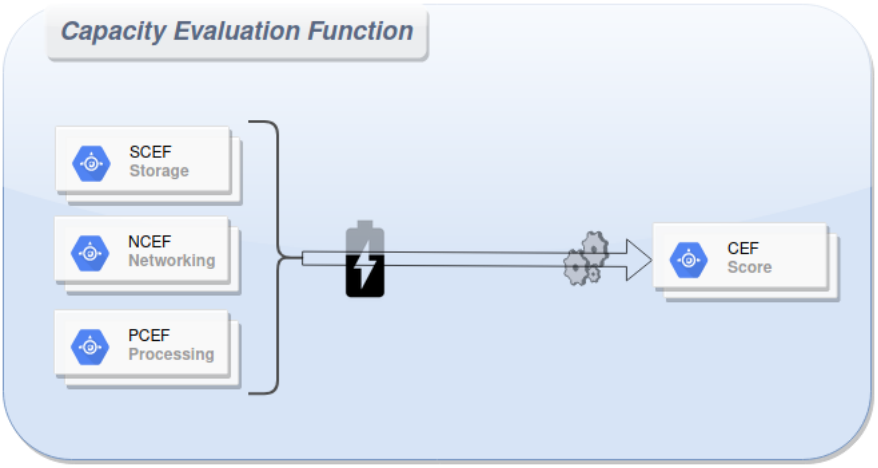


Figure 3.5: Capacity Evaluation Function score

groups. The larger the group is, the more related keys the KM will have to manage. Thus, the KM must have some some storage capabilities, especially if we want to preserve the scalability which can involve numerous group fellows. The SCEF was defined in [17] by:

$$c_k = pm \,.\, \frac{sc_k}{ks} \tag{3.1}$$

where $\begin{cases} s_k : storage\ capacity\ of\ a\ node\ u_k \\ sc_k : storage\ capability\ of\ a\ node\ u_k \\ pm : usable\ percentage\ of\ memory\ by\ protocol \\ ks : size\ of\ a\ key \end{cases}$

For conformity issues, we bring slight edit to the notations above, thus redefining the SCEF as:

$$s_k = pm \,.\, \frac{sc_k}{ks} \tag{3.2}$$

s_k : $storage\ capacity\ of\ a\ node\ u_k$

The rest of notations are maintained.

The next step is the definition of PCEF. The PCEF is a function we introduce in order to assess the processing capacities of the node, a potential KM candidate. Actually, all group related keys are exclusively generated by the KM. But a key generation involves the execution of cryptographic functions, which are by default expensive in terms of processing. As discussed in Section 2.3, lowering the standards of cryptographic systems implementation

may degrade its reliability. Thereby, enough processing capacities are crucial to maintain a high resilience level of the protocol. In Section 5.1.1 of [17] related to the theoretical analysis of the overheads of the KM, we cite:

> roperty 2: Calculation overhead on the KM is proportional to the sum $p + m_j$.
>
> (P)

Based upon this property, we define the PCEF score as follow:

$$p_k = pp \,.\, \frac{cc_k}{cs \,.\, (p + m_j)} \tag{3.3}$$

where $\begin{cases} p_k : processing\ capacity\ of\ a\ node\ u_k \\ cc_k : computation\ capability\ of\ a\ nod\ eu_k \\ pp : usable\ percentage\ of\ processor\ by\ protocol \\ cs : overhead\ of\ crypto\ system \\ p : number\ of\ subgroups\ of\ the\ group\ G \\ m_j : number\ of\ nodes\ in\ subgroup\ S_j \end{cases}$

The above definition of pk takes very well into consideration the fact that the KM will handle exchange security between other groups and his own, during different rekeying operations, according to the rekeying procedure defined in MGKMP. The function considers the percentage of processing the protocol can use. This means that nodes which processing units are perfectly designed to only handle their application related tasks, such as some wireless sensors, won't be very favorable as KM candidate. The computation capability *cc* is a static value based on physical characteristics of a device (eg. the processing unit's frequency in Mhz).

The last of the three capacity criteria to consider is the NCEF. Rekeying operations, as defined in MGKMP, involve big time communication overhead for the KM, from joining messages to leaving messages. Besides, theses messages are transmitted in all possible fashions: unicast, multicast or broadcast. Moreover, for a newly joining node, the first communication channel is to be established with the KM in order to proceed with the join (see Section 2.3.2). This channel has to be robust from a networking point of view, and shall not be hindered by low networking capabilities. As mentioned, rekeying operations might require the KM communicating with other groups, eventually physically far distanced, which even highlights further the importance of communication range and strength of the KM candidate. Even if the node is selected as supplicant KM, it will have the task of monitoring the main KM, which involves way a lot of networking. The deputy KM has to be ready to assume functions of an active KM at anytime anyway. In Section 5.1.1 of [17] related to the theoretical analysis of the KM's overheads, we cite:

roperty 1: Communication overhead on the KM is proportional to the sum $p + m_j$.

(P)

Based upon this property, we define the NCEF score as follow:

$$n_k = bw_k \,.\, \frac{rr_k}{(p + m_j) \;.\; \max(ms)} \tag{3.4}$$

where $\begin{cases} n_k : networking\ capacity\ of\ a\ node\ u_k \\ bw_k : bandwidth\ of\ u_k\ usable\ by\ the\ protocol \\ rr_k : radio\ range\ of\ u_k \\ ms : size\ of\ a\ message \\ p : number\ of\ subgroups\ of\ the\ group\ G \\ m_j : number\ of\ nodes\ in\ subgroup\ S_j \end{cases}$

Once again, the KM may be dealing with inter-group communications during rekeying operations, as defined in MGKMP. Therefore, the forehead definition of n_k takes the radio range of u_k into consideration, for practical usability concerns.

Following the three main technical criteria, one extra criterion is considered to ensure energy reliability of the KM. Both processing and foremost networking are energetically expensive. The energy correlation value is made up to adjust the final CEF score to favor the most green of nodes. It is defined as follow:

$$e_k = \frac{re_k}{ed_k \,.\, pu_k} \tag{3.5}$$

where $\begin{cases} e_k : energy\ attribute\ of\ a\ node\ u_k \\ re_k : residual\ energy\ of\ a\ node\ u_k \\ ed_k : energy\ drainage\ of\ u_k \\ pu_k : percentage\ of\ processor\ in\ use\ for\ u_k \end{cases}$

The function considers only remaining energy of the device by the time the election process is kicked off, then it's a dynamic value that evolves (mostly decreases) overtime. The device's total energy is not relevant. Energy drainage of u_k is actually a static value based on physical characteristics of a device. It could for instance the energy consumed in $J.s^{-1}$ during transmission or processing.

With all input values settled, we can finally compute the final CEF score as follow:

$$c_k = e_k \,.\, \left(w_s s_k + w_n n_k + w_p p_k\right) \tag{3.6}$$

where $\begin{cases} c_k : capacity\ score\ of\ a\ node\ u_k \\ w_s : storage\ capacity\ weight \\ w_n : networking\ capacity\ weight \\ w_p : processing\ capacity\ weight \end{cases}$

It's this very ck which will serve as score to assess the overall capacity of a node u_k, and serve as basis for the election process. Weight correlation values are first determined through simulations and experiments. Their values remain static afterwards, thus to limit computation overhead of the CEF and increase the protocol's efficiency.

The Run Threshold

We introduce the election run threshold, dubbed rt, as a static or dynamic value. In order for a node to run in an election, the node's CEF score has to be above this threshold. There is no formula for the value of rt. Since it depends very much on the used hardware and the use case application, it actually has be determined later on by simulations and experiments during the implementation of the protocol. Nevertheless, we consider by default that it's set to $rt = 0$. This threshold actually limits the number of participating nodes as candidates during the election. Therefore, it decreases the overall bandwidth consumption during the election, and improves the votes accuracy.

3.3.2 Operational Mode

This Section mostly describes the solution's algorithmic behavior. It defines different newly introduced entities and describes their roles as well.

The role of deputy KM

In addition to the active KM, we introduce a supplicant KM, also dubbed as deputy KM. In the following, the main CH will be referred as main KM, active KM or simply the KM. It has two main functionalities. First, it's the node to take over the active KM when needed. This can be upon the compromise or breakdown of the latter, or just for rotation. This need to have a well designated node ready to immediately take over is necessary to prevent a void in the head of the group. Such void in leadership is, just like in real life, a huge security vulnerability which leaves all the group's communications at risk. Second, the deputy KM can serve as a monitoring device for the active one. It would check over it regularly to verify it's neither inactive nor compromised. This check over message could be a simple ping

message, in case we just want to verify the main KM's availability. In case we want the check over to include both availability and integrity of the main KM, then the supplicant can just send a challenge to the KM. If the KM doesn't answer at all over a certain time lapse, then it's down. If the KM replies with a wrong answer, then it's compromised. In both situations, the deputy KM knows it must take matters into its own hands. Since the beginning of the protocol's definition, we often assume the presence of a security component in the network, in charge of monitoring availability and integrity of all network's nodes altogether. With this solution, we actually (i) make the protocol less dependant on external entities and ready-to-take assumptions, since the monitoring process is thereby partially internalised within the protocol. Thus, it makes the protocol more resilient. We also (ii) manage to decentralise the KM's failure detection and recovery, since we have one deputy KM per group to take care of it. Thus, it makes the protocol more scalable.

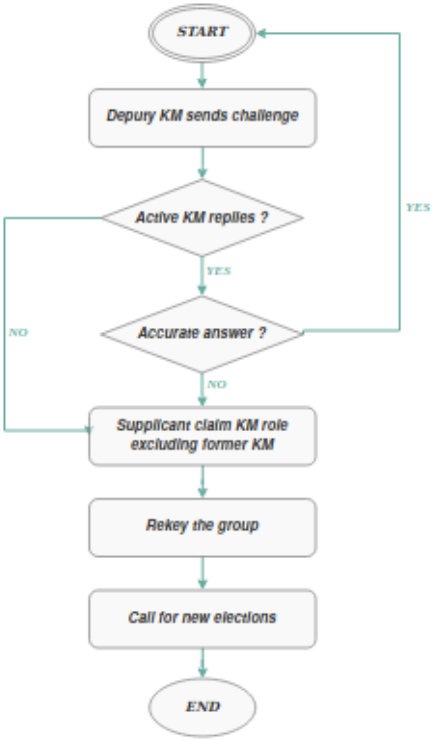


Figure 3.6: Failure recovery workflow

Election mutual exclusion and roles definition

The protocol's design assumes that a node can belong to two distinct groups in the same time. Basically, a node cannot be the KM for more than one group. Nodes dispose of a binary parameter *elected*, by default set to *False*. If a node gets elected in the context of any group it belongs to, this parameter is switched to *True*. If a node n_i is affected to the two distinct groups G_1 and G_2, while also been elected KM for G_1, then it definetly is not and will not be the KM for G_2. If some election takes place in the context of G_2, n_i will have its parameter elected already set to *True*. Hence, it just won't broadcast its CEF score, and so won't be promoted as a potential candidate. But this restriction doesn't prevent the node from participating in the voting process.

In case a node other than the KM is compromised, the KM has the charge of the rekeying operation, as originally defined in MGKMP. Upon a node's join or leave, the procedure remains the same as well. However, if it's the KM itself which is detected as compromised or flooded with a huge charge of tasks, then a failure recovery process is instigated. When in place, the detection is done by the deputy KM.

To enhance the scheme's performance and strengthen its decentralised side, we introduce two operational optimisations regarding tasks mainly handled by the KM. One concerns the key share during rekeying process, while the other is more about its very own generation. During a rekeying operation, the KM is responsible for new keys generation as mentioned. These new keys shall be communicated to the concerned group members. For this, the KM doesn't necessarily have to deliver it to every node one by one. Some fellow members might already have pre-established communication channels between them, encrypted with keys known to only both of them. To save its energy and bandwidth, the KM might rely on other fellows, having low energy-consuming tasks, to relay the information in a peers-seeders fashion. Besides, upon a new node's join, the KM doesn't necessarily have to generate new keys for the rekeying operation. It could actually make use of prefix encryption and generate derived keys from the old ones. Nodes detaining the old keys, those are group fellows, are able to guess the new ones. But the joining node, receiving the new keys, won't be able to climb up to the old ones, which satisfies the forward secrecy requirement. This however is applicable only to node join operation. By definition, this prefix encryption enhancement cannot work out for node leave or eviction operations.

Inter-group communications management

In the reviewed litterature (Section 1.1.2), election-based schemes assume a CH for each group. Since proposed solutions are aimed at addressing routing problems, groups were not supposed to communicate between each other. For instance, each group has its own KM, and the latter shall mind

only his groups internal business. Therefore, they manage only intra-group communication issues. However, the MGKMP leaves the door open for inter-group communications. In the original centralized scheme, the problem wasn't even to worry about since we have one central KM for the whole network. In order to solve this issue, our solution suggest that every KM within the network is capable of operating on the inter-group level, and it's the situation that determines which one in particular to do it. The design of MGKMP assumes that nodes hold DEKs and KEKs to secure group communications on the service level. These keys are very important, because they're used by nodes in order to enhance the protocol's scalability and efficiency. Nevertheless, a group by definition is a unique combination of service(s). Hence, These mentioned keys actually involve communications on the inter-group level, which we will refer as the federal level. It's right here where the limit of previously proposed works on CH schemes bring front another limitation: if we have a KM for every group, such is our case, which entity is responsible for tasks involving more than one group ? If we look deeper these so-called federal processes, they mainly interfere during rekeying operations, the KM's main task. So basically, the renewing of these service related keys occurs in the aftermath of a group's membership update (node's join, leave or eviction). Therefore, we assume that it's the KM for the concerned which will handle the whole task. It will take of renewing the group-related keys and all the other keys requiring renewing. Then, it will communicate these keys to the concerned groups if necessary. This way, any locally elected KM can actually be the federal KM, and may assume these double role whenever the situation requires it. Handling two positions simultaneously and for a short time slot makes this solution efficient. Allowing all so-called local KMs to do it makes the inter-group key management decentralised, and thus, the solution scalable and compatible with heterogeneous networks.

3.3.3 Election process

This section describes the election process. It defines notations related to the operation, and enumerate its different steps.

Group Key

Since the election takes place in the context of a one particular group, the election procedure detailed in the forthcoming paragraph would actually require broadcast operations within this group. These transmission should be encrypted on the group level. However, the MGKMP as defined do not introduce such key, probably because it's not actually needed for its main purpose, rekeying operations. Groups were basically designed to be logical and completely transparent to the application. As for Key Management and

rekeying processes, we can do with just service keys and subgroups keys, no need for group keys. Thus, in order to secure communications with in a group without limiting ourselves to subgroup constraints, we have one existing solution. Since a group is a unique combination of services, we could secure the message by encrypting over with all relevant service keys. All nodes belonging to the group are, by definition, holders of necessary keys to decrypt the message. Nevertheless, this solution turns out to be very expensive in terms of processing. Therefore, we introduce in this section a Group Key. We denote K_i the Group Key of the i^{th} group, held by all group members. This assumption increases very slightly the storage overhead for nodes. But, it saves much more in the processing overhead, and thereby less energy consuming.

Notations and messages glossary

$$EM1 : CH \Rightarrow G_i : \langle\ \varnothing\ \rangle_{K_i} \tag{3.7}$$

a call for election broadcasted by either current KM or its deputy

$$EM2 : u_k \Rightarrow G_i : \langle\ c_k\ \rangle_{K_i} \tag{3.8}$$

CEF score broadcast by potential candidate

$$EM3 : u_k \Rightarrow G_i : \langle\ m,\ n\ \rangle_{K_i} \tag{3.9}$$

vote broadcast by elector node, where u_m and u_n are the voted KM and supplicant respectively

$$EM4 : u_m \Rightarrow G_i : \langle\ \varnothing\ \rangle_{K_i} \tag{3.10}$$

KM role claim broadcast

$$EM4_{bis} : u_n \Rightarrow G_i : \langle\ \varnothing\ \rangle_{K_i} \tag{3.11}$$

supplicant role claim broadcast

Fig. 3.7 demonstrates the assumed sender of each message. The timeline order of sending these messages goes from $EM1$ to $EM4/EM4_bis$. As shown in the figure, all message are actually broadcasted by their source nodes to the whole group.

Election procedure

The choice of the KM and eventually its deputy is based on an election procedure, during which nodes will vote for the most reliable KM. The procedure's outcome must satisfies the election of a technically reliable and secure KM, while consuming the minimum of bandwidth.

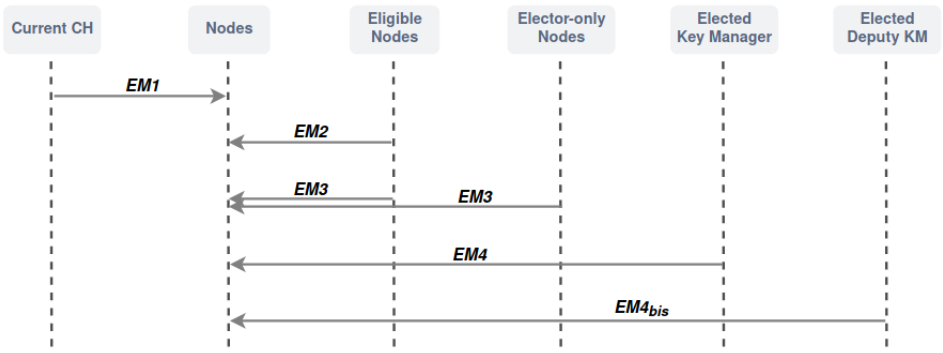


Figure 3.7: Election process message exchange

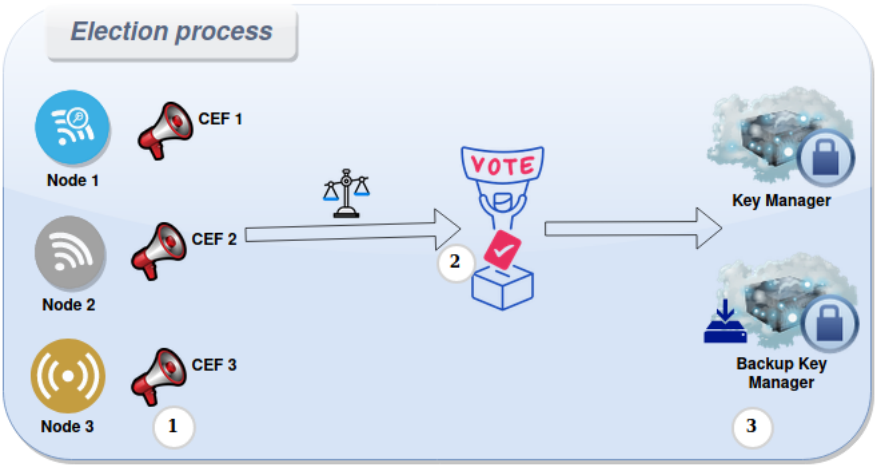


Figure 3.8: Election procedure

When an election trigger is pulled, the current CH, whether KM or supplicant, will instigate the election procedure. It first broadcasts the message $EM1$ to the relevant group. This message is a call group members for elections. Once group fellows receive this message, each potential candidates will update their CEF score. These nodes must definitely have their *elected* parameter set to *False*. Nodes which *elected* parameter is set to *True*, are deemed not to run and would skip this step. For each running node, if its updated CEF score is higher than the rt, it then braodcasts the message $EM2$ to group members. This message contains the updated CEF score and the identifier of its sending node, for others to benchmark to most reliable candidate. Followingly, nodes will receive different capacity scores from different candidates. Thereby, ther dispose of a list of scores corresponding to group fellows, including their own if they're candidates. Every node will benchmark scores in its list in order to choose the highest two. Then it will broadcast the message $EM3$, which contains identifiers of both chosen candi-

dates for KM and supplicant roles respectively. Now every candidate node receiving this message will keep a count of which node received most votes. So theoretically, the count should be the same for every node, with the exact number of votes for the same couple of KM/supplicant, knowing who the ones are. However, due to the probabilistic aspect of networking and some nodes IoT-related liabilities, it doesn't always work out perfectly as expected. That's why we are keeping track of this count. If a candidate node sees that it has been elected for either role, it has to claim it. In case the node is elected as KM, it will broadcast the message $EM4$ to the group, wich contains its identifier and useful coordinates. This message announces to the group that the sending node is the new KM. In case the node is elected as supplicant to the KM, it will broadcast the message $EM4_bis$ to the group, which contains its identifier. This message announces to the group that the sending node is the new deputy KM. Now this is concerning candidate nodes. For the rest of elector-only nodes, they actually go into a stall mode since sending message $EM3$. They don't need to neither receive other nodes $EM3$ messages, nor keep track of the election progress. This will save them significant networking and processing overhead, and thus energy. For every single node, the reception of both claim messages marks the endpoint of the election procedure. This notification could be relayed through different node hops, just like the new key sharing. This helps reducing the networking overhead for the KM and make sure every node received the notifications.

Election outcome

The election's purpose is to efficiently benchmark different nodes CEF scores, in order to select a KM and a deputy for the group. The election's outcome is actually this KM/deputy tuple, which once the election procedure is through, they should be getting their new status. Now at the very beginning, the election had been called for by either the former KM or the former supplicant. This so-called election instigator could itself be re-elected to one of the two positions. In case the KM is the one calling for elections, there is no way it would be re-elected in either positions. Actually, this case matches only a KM rotation due to its drainage. The concerned node wouldn't be participating in the election procedure in the first place, so this case is irrelevant. However, if the election's instigator is the supplicant, it would take part in the elections as a candidate, and has pretty good chances to be re-elected.

Election conflicts

An eligible node, belonging to multiple groups, can participate in two election processes simultaneously. But if it gets elected in two distinct election process simultaneously, it shall assume the responsibilities of a KM where it's more needed. For this purpose, a difference in the CEF score with the

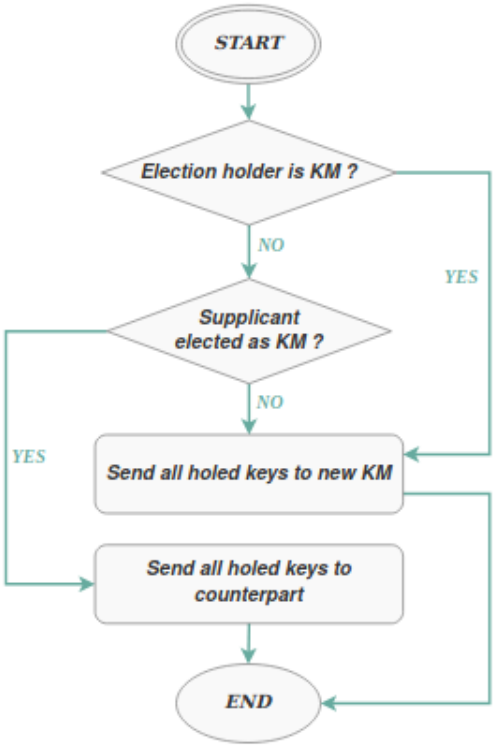


Figure 3.9: Election outcome

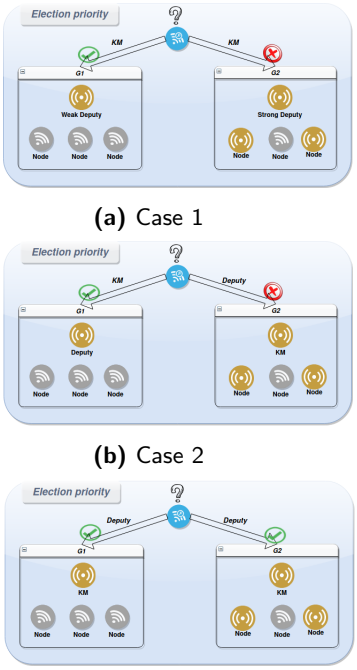
corresponding elected supplicant is measured. And where the difference is less significant, the node can decline the position to accept the other one. In other words, the choice goes to the group where the capability gap is wider in order to avoid under-performance issues. However, when a node is approved as a KM in a group's election process, and as a supplicant in another, it would be accepting the KM position and declining the supplicant role. Nevertheless, if a node is simultaneously approved for supplicant in two elections, it can assume both as the supplicant role doesn't have consequent heavy assignments.

For the following detailed example, we introduce the distance measurement:

$$d\left(n_1, n_2\right) = e_k \,.\, \left(w_s s_k + w_n n_k + w_p p_k\right) \tag{3.12}$$

where $\begin{cases} c_k : capacity\ score\ of\ a\ node\ u_k \\ w_s : storage\ capacity\ weight \\ w_n : networking\ capacity\ weight \end{cases}$

Let n_i a technically capable node belonging to the groups G_1 and G_2, with both groups having an ongoing election process each. Let ${n_i}^1$ and ${n_i}^2$ also



(a) Case 1

(b) Case 2

(c) Case 3

Figure 3.10: Election conflicts

technically capable nodes belonging to G_1 and G_2 respectively. If parameter elected for n_i is off, then it can broadcast its CEF score within the context of both groups as a potential candidate. If it receives an approval for the KM in a G_1 related election with ${n_i}^1$ as supplicant, it also receives approval as KM in the G_2 related election process with ${n_i}^2$ as supplicant, an arbitration should take place. Since the election is based on the CEF score, ni will measure the differences d_1 and d_2 with the node ${n_i}^1$ and ${n_i}^2$ respectively. n_i will assume the charge of KM for the group related to the corresponding node with $m = max\,(d_1,\ d_2)$. Hence, it will decline the election approval for the other group, letting its corresponding deputy take over.

Fig. 3.10 illustrates the participation of the blue node in two separate elections with in the context of two different groups. The figure demonstrates how to proceed in case the blue node was elected as either KM or supplicant in both elections simultaneously. Table 3.1 sums the three cases up.

3.3.4 Failure recovery process

There are three situations which could be qualified as failures requiring the instigation of a recovering process. In the first one, the KM is somehow unavailable, due to some flooding or a technical breakdown. The second situation concerns an attack leading to the compromise of the KM. And

Table 3.1: Election conflicts

		G_1 **related election**	
		Key Manager	***Supplicant***
G_2 **related election**	**Key Manager**	verboten	verboten
G_2 **related election**	**Supplicant**	verboten	allowed

the last one is when a node is overworked and willing to hand over its responsibilities.

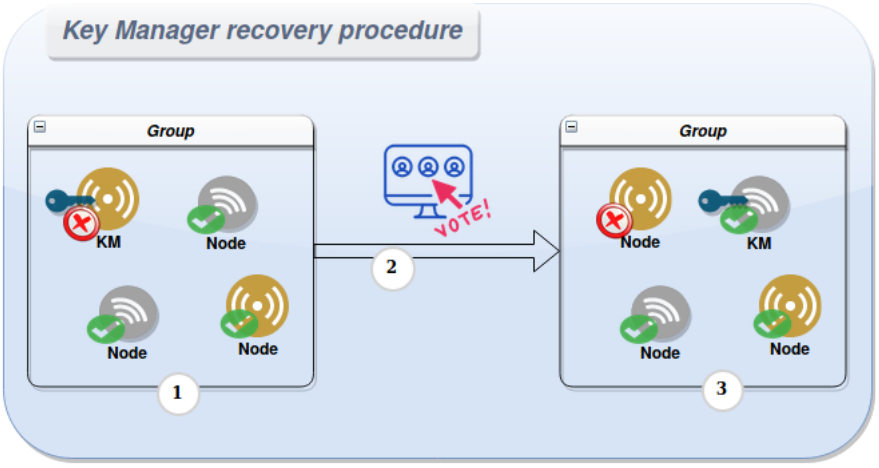


Figure 3.11: Key Manager recovery routine

Starting with the first situation, the KM might be subject to overworking due to the important networking and processing overhead it has to bare. This could also be caused by veiled activities. Unless it has specific procedures to curb this burden, it obviously makes it vulnerable against breakdown. When the unwanted occurs, it's first detected by the deputy KM, thanks to the regular check over procedure. Since the active KM will be unable to answer its deputy's requests, this will let the latter know it's technically down. From here, the supplicant broadcasts a claimer message to the group announcing that it's taking over the leadership of the group. The former KM is excluded from the broadcast though. Then it starts a rekeying operation of the group. An finally, it calls for a new election to redefine who is the KM and his deputy on solid basis. We actually have no idea whether the breakdown is due to technical dysfunction or malicious attack such as DDoS. It could even be compromised by a malware, which prevents the reply upon check over to make it look like a normal breakdown. Keeping the former KM in the loop is exactly taking the risk of having a potential Trojan horse inside

its network, without bothering taking any precautions. This significantly increases the resilience of the protocol.

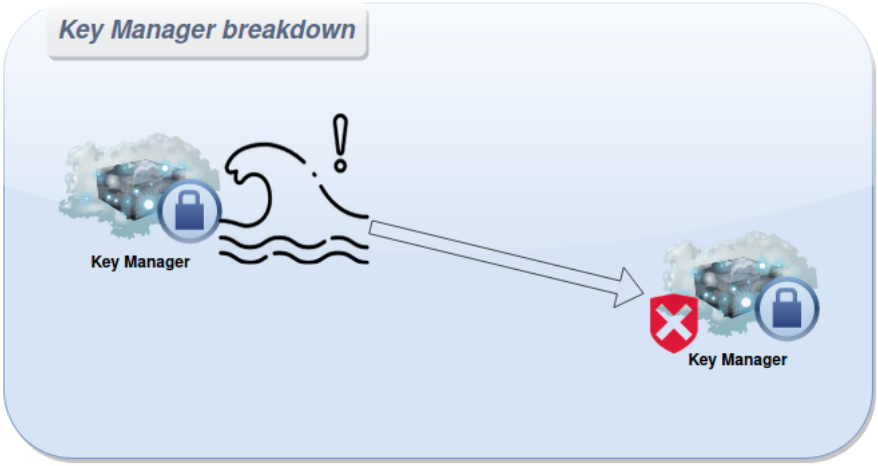


Figure 3.12: Key Manager breakdown

For the second situation, either we assume that we have a security entity (SE) physically separate from the KM (an IDS for example). It's this component that guarantees the detection of eventual cyber attacks, and therefore, the instigation of the recovery process. Or, when in place, we can use the supplicant as a controller to detect the compromise, and so launch the recovery process. In the first case, we don't consider the detection phase since it's totally handeled by the SE. But once the dysfunction is detected, the SE report directly to the KM deputy. In the second case, it's the deputy KM that detects the compromise of its leader. The scenario would start with the supplicant receiving an inaccurate reply from the KM, upon a regular check over request. The inaccurate reply is basically a wrong answer to the deputy's challenge. In both cases, the latter broadcasts then a disclaimer message to the concerned group fellows announcing that he is the new KM, while excluding the former one from the broadcast. It starts a rekeying operation after that of the group, since here we know for sure we have a node (the former KM) to evict. And finally, it calls for elections to settle a new appropriate KM, or at least fill in the vacant position of the supplicant.

In this last situation, the KM isn't necessarily suffering from any particular malfunctioning. For several reasons, it may choose to hand over its responsibilities to another node. It actually foresees some situation where, for application reasons, the KM will have additional expensive tasks to handle besides the Key Management ones, exceeding its capacities. The step off cause might simply be the KM leaving the group too as part of the application. Instead of forwarding the task to the deputy, it first calls for anticipated

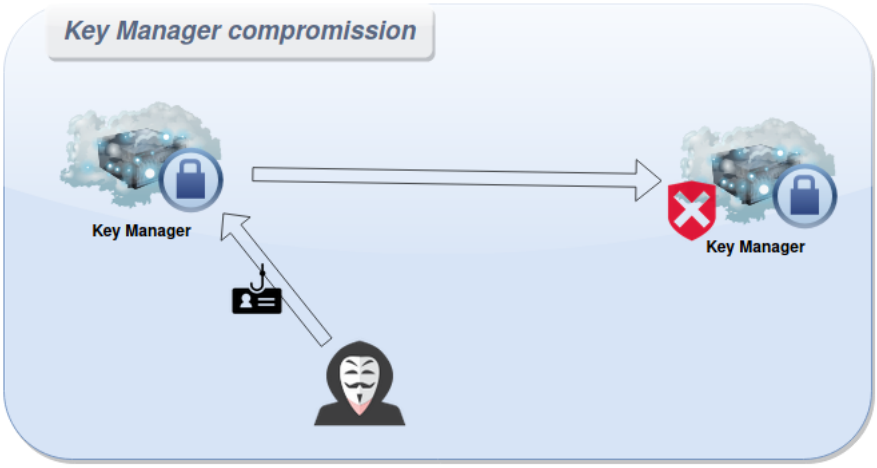


Figure 3.13: Key Manager compromise

election in which it would not be a candidate. The election will output a newly KM . Only then, the desisting KM hand over the role to the latter.

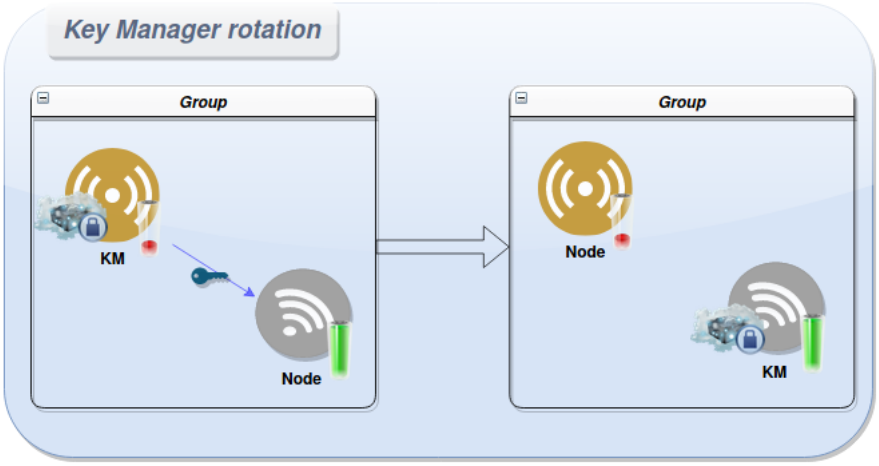


Figure 3.14: Key Manager turnover

The scheme, as explained, introduces a supplicant which monitors the active KM and ready to take over in need, while actually allowing any node to be the KM itself. This alongside the performance optimisation introduced for new key sharing, are made up to address the issue of ***single point of charge***. With in the same progress line, the capability-based election process concatenated with the failure recovering processes, their conditions and the supplicant's monitoring are actually made up to address the issue of ***single point of trust***. This way, the overall election-based scheme solves the issue of ***single point of failure***, by locking on the different reasons which led to

this problem in the first place, as well as its unexpected consequences.

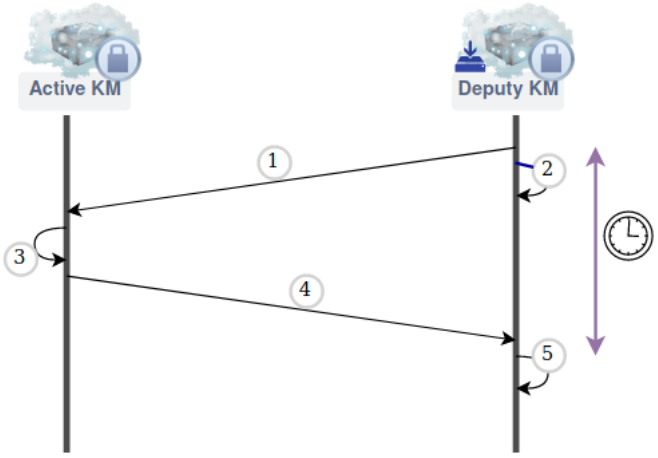


Figure 3.15: Check over message exchange

Fig. 3.15 illustrates the message exchange for a routine check over procedure. Above labeled steps correspond to:

(1) Challenge request

(2) Set timer on

(3) Answer computation

(4) Challenge reply

(5) Set timer off + Challenge validation

The challenge validation implies both, a correct answer to the request and a response time within the set timeout.

An alternative routine would be the supplicant listening to the active KM, whereas the latter sends regular messages to its deputy announcing it's still up. For integrity consolidation these messages have to be cryptographically signed. This pulse-check routine reduces the networking overhead by a factor of two. Although from a security point of view, it's much less reliable than the check-over routine previously described.

In order to combine advantages of both routines, we define a hybrid solution, where regular checks are performed every time lapse, referred as Δt. These checks are mostly simple pulse-checks. But every certain number of time slots Δt, referred as *nc*, we perform the double check-over routine.

This way, we ensure availability and integrity of the KM, while significantly reducing the networking and processing overhead for both parties. Δt and nc are settable and configurable parameters, since depending on the actual

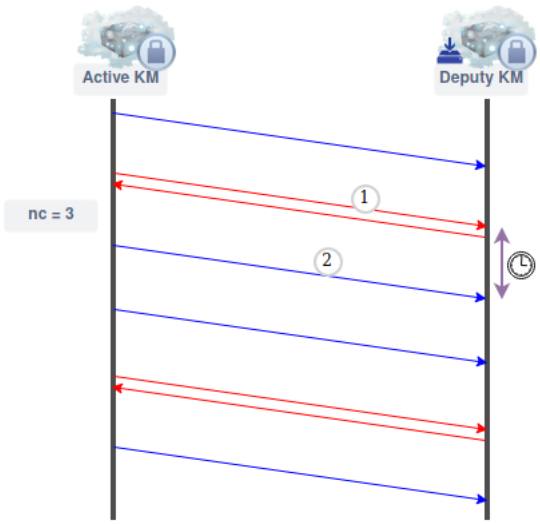


Figure 3.16: Check Over routine

use case, the balance between performance and security might flip to either side. If we want to keep only the first routine, one can just set nc to $nc = null$. The bigger Δt is, the smaller the networking overhead is. Fig. 3.16 illustrates this hybrid solution, with:

(1) Double-way check-over routine

(2) One-way check-over routine

3.3.5 Security analysis

The CEF score is intentionally designed to be an average value totally blending regarding the input values. It's actually more secure if the CEF score doesn't leak any data about a node's real capabilities in terms of networking, storage and processing. Knowledge of some physical properties of a device can be used to mount cryptanalytical side-channel attacks. During an election process, any group member node u shall be able to capture CEF scores of other fellow nodes. But it won't need these information as long as it's a fellow member of the group and holds required keys to decipher transiting communications. When u leaves the group or gets evicted, the MGKM protocol relies on the group rekeying in order to guarantee forward secrecy. The recently leaving (or evicted) node u can seek to decipher group's communications based on its previously acquired knowledge on communicating entities. This way, forward secrecy is put at risk. This also applies for a recently joining node v. During the first election process after its join, different nodes will be broadcasting their respective CEF scores, which express their respective physical capacities. The node v might then use these

acquired scores to carry out side-channel cryptanalysis attacks on encrypted messages collected before its join. This way, backward secrecy is put at risk. Since the CEF's output is a single dark value, it will therefore prevent these category of cryptanalysis attacks. Hence, our solution maintains forward and backward secrecy properties.

When a node u acting as KM is compromised and evicted, the supplicant immediately takes over the group leadership and rekeys the group, while excluding the former KM from this rekeying process. Only afterwards it calls for an election and kicks the election process off. Messages related to the election which would take place will be encrypted using a newly generated group key. By applying the forward secrecy principle guaranteed by the rekeying process, the former compromised KM will hence be unable to decipher them. The group's communications are then secure.

Another security issue had to be considered when designing the check over process. As previously hinted, the simple check over routine utulizes cryptographic message signature only as a security measure to authenticate the KM. However, the double check over routine utulizes challenge-response authentification which can be (but not necesserily) coupled with cryptographic message signature as security measures. The latter is obviously more secure, but less performing. The combination of how all of this will be implemented is left to the eventual R&D engineer who might take over the protocol. What actually interests us here is the foreseen security cradle put in place for the overall check over process. Now both considered measures have their own advantages and drawbacks, with the challenge-response authentication slightly safer. Nevertheless, none of both can be taken for granted. On one hand, new attacks on both keep surfacing constantly. On the other hand, both are hardened and enhanced continuously. Besides, these measures themselves bring their own parameters and vulnerabilities. As seen in Section 2.3.3, the choice of message authentication codes and signature algorithms can have a huge security impact. In the same line, different challenge-response protocols for different situations exist. The engineer implementing the process knows his own use case and security risks. Therefore, he knows better which combination of all this fits best.

In short, the check over process intentionally embeds two different and combinable security measures. The objective is to offer a maximum of options for different security situations.

3.3.6 Discussion

The following discussion brings nothing new in term if technical specification. But since this a fundamental research work, it's more destined, but not only, to any eventual engineer who might be implementing the solution. This paragraph actually tries to give a glimpse on how practical the solution

is and what kind of asset and disadvantage the above technical description can have on the implemented system.

Flexibility and adaptability

The main drawback of this scheme, is the networking overhead that comes with. Despite the suggested optimisations, we still have a high number of required messages to undergo related procedures and processes. In order to get over this lack, the scheme should be designed in a flexible way.

For the protocol's operational mode, the CH rotation has to be implemented carefully, and sometimes, even set as optional. This could be very well relevant to some applications where the KM's choice is static and permanent (eg. Cloud server). In this case, all message exchange relevant to CH rotation is probably useless and a waste of bandwidth and energy.

Besides, the supplicant role should be optional. We can't be always sure whether we will dispose of enough qualified candidates for both roles. And sometimes, the infrastructure could designed in a way that one particular component is intended to be the KM for his group. In which case, the engineers building the infrastructure could have foreseen alternative measures to sort out any failure in the KM. Furthermore, the supplicant plays also a security role as it monitors the KM's availability. Depending on the global infrastructure and the KM's ecosystem, this role could be obsolete since there is some other component doing it. In the same line of thought, this so-called powerful KM could foreseen by its implementing engineers to cover more than one group. Thereby, the mutual exclusion in elections should be optional as well, in particular, disabled. In this direction, it seems at first we're heading back towards the original centralised KM scheme. Nevertheless, the idea of the election-based scheme is to tackle the issue of single point of failure. But the last call is left to the engineer implementing the protocol, so that it offers the protection it claim when necessary. Besides, the scheme introduces itself as a compromise between centralised (unique KM) and fully decentralised (Blockchain) schemes. This way, among n groups, we can have one or some of them managed by one capable KM, technically just by disabling an option, while the rest of the groups dispose of their own elected KM. So here the scheme doesn't cover the whole infrastructure, but it's used right where it's needed to tackle the central problem previously defined. And this is particularly useful for heterogeneous networks. This also facilitates migration from centralised solution to this one.

As for the election process, the elected parameter blocking mode has to be implemented in a way it can only be enabled manually. The feature should basically be optional. This feature is useful in some cases where we might have some very battery-weak devices. Despite the set eligibility threshold, just the computation of the CEF and the preliminaries could be,

for some devices, enough for the battery to be rolled out. Nevertheless, this option isn't pertinent for all IoT use cases. And since it brings in restrictive measures for concerned nodes, then it should definitely be configurable only manually depending on the application.

The eligibility definitely cannot be stiff. In order to improve the scalability and adaptability of the protocol regarding heterogeneity, CEF threshold has to be optional and configurable as well. Depending on the application, we can have very different scales of devices power. Thereby, the very definition of a powerful node would change. Hence, we shall be able able to lower down our standards, even sinking it til 0 if necessary. On the other side, we could be willing to top it up, if the engineer implementing is sure about the robustness of his devices and his need for a very reliable KM. This CEF feature might also be left open for more input criteria (or less) depending on the use case. Nevertheless, it has always to remain as static as possible and least possible dependant on dynamic inputs. The latter actually can increase the turn over rate for the score's update, and so, an increase in the computation overhead.

Concerning the overall architecture of the protocol, it actually relies on one KM assigned to one group basically. This should be extensible for other layers of the infrastructure, such as having one KM per service or per subgroup. Such architectures and configurations have to be thought over and studied in depth, in order to conclude on their utility, efficiency and feasibility.

Inter-group communications management

Another approach would be based on a double layer election. The first defined KM election would be regarded as a local group election. We introduce a federal election which takes place between different local KMs, in order to elect a federal KM. The latter would then assume a double role, (i) managing his own group's keys and (ii) handling inter-group related keys. The local KMs are the only eligible candidates, and in the same time, the only voters. The rest of the election process goes the same as for the local one. This approach has the advantage that it bypasses all possible management conflict which might pop up according to the first suggestion. Typically, this might occur when the rekeying process concerns a node belonging to different groups sharing the same service. Hereby, more than one KM is supposed to handle the service keys, making a part of the process redundant between different concerned KMs. A drawback however shines when it comes to profitability. This approach is less scalable and efficient than the previous one.

3.4 Simulations and results

A simulator of the MGKMP was developed as part of the PhD thesis [16] anterior to this work. After studying the code source, the program was actually implemented to simulate the protocol's rekeying process and sub-grouping algorithms. Thus it's not fit for this use case. Working on this simulator to adapt it was considered indeed. However, this would have taken much more time than implementing a new one dedicated to the ongoing situation. The second solution was picked, because the simulation phase came at a time when the conference deadline for paper submission was closing in.

3.4.1 Simulator implementation

Two simulators were implemented with different levels of success. The programming language of choice for both is *Python3*. The first is much more complete and realistic, but has a lot of bugs. The second one is simpler, but still produced some decent results. The development approach adopted is object-oriented.

First of all, I thought of implementing a simulator, which will simulate the whole MGKMP architecture. In other terms, the simulator actually implements possible entities such as nodes, KM, groups and subgroups. The communication between different nodes takes place via network sockets. The simulator uses the *select* function from the *Socket API* to handle connections for every node. The simulated algorithms mainly include those related to the election-based scheme, ie. the election and check-over processes. The messages exchanged were also implemented to be encrypted, to provide the most realistic environment possible. This simulator is pretty sophisticated, so its implementation requires several functions management and consequent debugging. Due to time constraints, it couldn't be finished. So I switched to a similar simulator, but with some features cut off.

The second simulator doesn't emulate the MGKMP architecture entirely. It only considers one group with several nodes, two of which are KM and Supplicant respectively. It does implement a messaging structure. But communications and encryption are actually abstracted, they go through the *Simulator* object. The latter acts as the orchestrator running the show. It coordinates exchanges between different nodes and calls election related algorithms when needed. So this simulator is only good enough to emulate the algorithmic behavior of the developed solution, as well as the influence of different parameters variation.

The *Node* object has two different categories of attributes. First, it has physical capabilities such as radio range and processing capacity. Those values are random, attributed at the simulation's start and remain unchanged all along. They are different for every node. Second, it also has volatile at-

tributes which tend to evolve overtime, such as the percentage of processor in use and residual energy. These attributes start with the same values for all nodes. For example, the starting residual energy for all nodes is set to $re = 1000$, which is the maximum. Then it decreases overtime according to the simulation's events at a different pace for each node. Capabilities values aren't realistic, but rather based on a scale in such a way to have credible comparisons.

In order to generate results, the main program is implemented in *results.py*. It starts a simulation according to given parameters (Δt, nc, rt ...). Then, it stimulates different processes such as election or check-over routines. On every new event, it takes snaps, which mean it gathers information about all nodes and simulation status and stores them in a file. Based on these files, the program can generate graphics using the *matplotlib* library.

3.4.2 Simulation results

In the following examples, we assume that a node which was previously elected KM cannot be re-elected. Thus to measure the influence of KM turnover on the overall organization.

The simulator initiates 8 nodes at first, none of which is KM nor Supplicant. These nodes are attributed random physical capacities as mentioned, so they can compute their initial CEF score respectively. The initial status is saved to *snap00*.

In the first scenario (see Appendix A), the simulator instigates a first election, during which *Node 6* is elected KM and *Node 7* is elected Supplicant. Then it instigates a second election, during which *Node 7* is elected KM and *Node 1* is elected Supplicant. Fig. 3.17 illustrates the energy and CEF score evolution of *Node 7*. Between $t = 0$ and $t = 3$, the node acted as Supplicant and its energy is hence decreasing. The second election took place between $t = 3$ and $t = 6$, after which *Node 7* became KM. What the figure illustrates, is that during the same time lapse ($\Delta t = 3$), the election process if far more exhausting than the check-over.

In as a second scenario, the simulation tries to assess the influence of nc value on the solution. The election process is a random event. Hence, I concentrated more on the check over process, since I have more control on its occurrence frequency.

Let's assume that a check process, whether simple or double, is an atomic operation. We fix the time gap between two consecutive checks to $\Delta t = 2$, and the simulation duration is fixed on 36 time units. The simulation consists in running the check-over routine over the simulation duration straight with different values of nc, and measure the energy consumption for both, the KM and its deputy. The nc value range goes from 1 to 30. Intuitively,

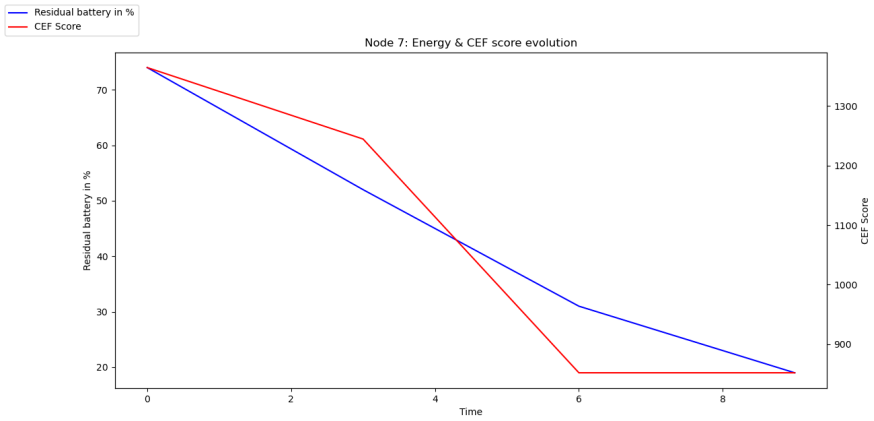


Figure 3.17: Node 7: Energy & CEF score evolution

a decrease in consumed energy during the simulation is expected as nc is increased, since a one-way check consumes less than a double one. From an energy perspective, we tend to increase this value. From a security perspective, we tend to decrease it. The objective is to find a satisfactory compromise.

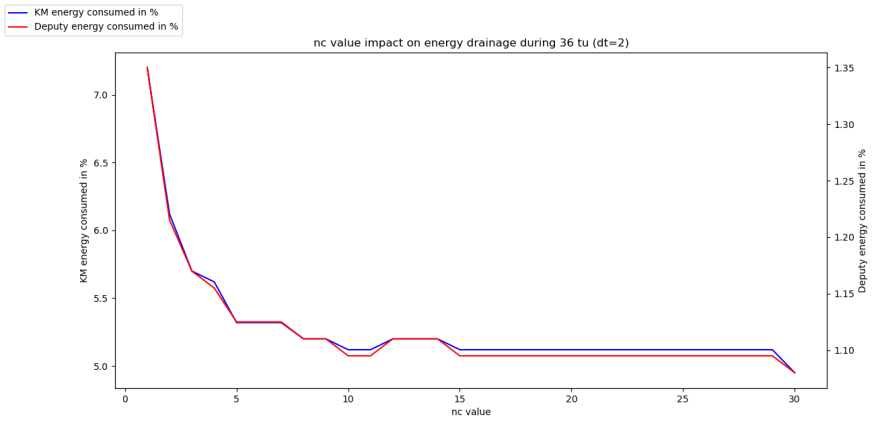


Figure 3.18: Variation of energy drainage according to nc value

Fig. 3.18 shows that this compromise can be quickly found, and it's valid for our two measured entities. The gain in energy grows pretty fast as we increase nc value between $nc = 5$ and $nc = 10$. Therefore, our algorithm is able to reach good energy performances, while keeping a certain security level ensuring the KM's integrity. There is a slight illogical increase in energy between $\Delta t = 12$ and $\Delta t = 14$ though. The reason isn't clear, but it could be explained by the simulator's random behavior.

It's ***very important*** to note that interpretations rely only on figures shape, regardless of the scales or values. As said before, the program simulates the algorithmic behavior of protocol. So we know that in a more realistic experiment, we would still get the same figure's shape, but the output values can be different. For example, Fig. 3.18 shows that between $nc = 1$ and $nc = 5$, energy drainage dropped by approximately 1.9%. It's this kind of value which is irrelevant, because physical capabilities are randomly attributed and disconnected with the physical realm.

3.5 Future work

If I pick up where I last left off, it is clear that simulations and experiments need to be pushed way further. First, simulations remain in their early stages. More tests should be carried out using a more sophisticated and realistic simulator. Second, real laboratory experimentation has to be carried out to observe the divergence between simulations and experiments, and eventually address the gap. The simulator itself isn't fully implemented. From a functional point of view, many features are poorly developed. For instance, a node can have a negative battery value, absolutely mindblowing. That's because until this point, these features aren't required to retrieve the needed measures. But as the simulations get more advanced, it's better to have something more complete. From a structural point of view, it's also better to make the simulations more realistic, regarding how nodes are organized and communicating. This actually goes back to finishing the second simulator which didn't work out in this internship.

Another task to look closer concerns the algorithms complexity. In this chapter, no such mathematical study was made, and yet it's important to assess. It was shown through simulations that election and check-over algorithms don't have the same enrgy consumption for example. A more more profound algorithmic study can reveal the exact difference, and eventually contribute in the complexity optimization.

Moreover, all issues described in Chapter 2 need to be treated the same way as the Key Manager. In particular, the work on the CEF score done in this internship can be intercrossed with the subrouping sequences (Section 2.2.3) and *mc* values.

Chapter 4

IoT Security engineering

This Chapter treats IoT security from an industrial engineering perspective. At this point of the internship, it was decided to cut the research work off, and turn more towards practical subjects. This marks the end of *Stage 1* and the start of *Stage 2*.

4.1 Threats landscape in the IoT

4.1.1 Main threats

As for all information systems, one of the most common threats IoT devices are facing is malwares. In difference with usual malwares we deal with on computer systems, IoT malwares are very often cross-architecture, because IoT devices themselves are based on different physical and processor architectures. That's to say that antagonists are adapting their malicious payloads to the heterogeneity in IoT. Besides the damage a malware can inflict on the system by definition, the cross-architecture side of IoT malwares make them very difficult to detect and classify. Moreover, energy-constrained devices are not likely to support complex malicious code detection systems. Alhanahnah et al. [8] made a step towards proposing a signature generation for cross-architecture IoT malwares. However, the IoT malware detection problem remains very much unsolved yet.

With over 40 billions connected objects reached by 2025, IoT devices are a very seductive tool for botnets and DDoS attacks. Going back to what has just been evoqued, since some IoT devices can be easily infected with malwares in a stealthy fashion, a hacker or a hackers group can then seek to infect a large numbers of objects in order to build a botnet. IoT botnets can be used to compromise heavily secured computer systems using their powerful brute-force potential. They can also be used to conduct some of the most massive DDoS attacks, such as the Dyn attack on major DNS providers

carried out by the Mirai botnet in 2016 [3]. The Dyn attack was the largest DDoS attack ever seen at its time with an unprecedented flood of 1 Tbps targeting OVH servers. Its record has only been broken by the attack on AWS servers in June 2020, with a record charge of 2.3 Tbps [11]. Once again, the attack was carried out by an IoT botnet. The exponential growth of IoT connected objects is only democratising these kind of lethal attacks, and posing more serious security challenges in IoT networks security. Molesky and Cameron [9] proposed a solution based on the "White Worm" technology, which uses the very technical basis of this threat and turn it up into an asset used for benign purposes. The scheme remains pretty theoretical though, and a lot is yet to be done to bring the concept into a realistic application.

Another major threat in IoT is data privacy, and this concerns some of the most common IoT applications. For example, IoT connected objects are more and more used in healthcare and smart homes. In those kind of applications, we usually have wireless sensors and other types of objects gathering and processing intimate (eg. smart homes) and sensitive (eg. healthcare) data about individual's. Therefore, IoT communications are a top priority target for network attacks, generally conducted by actors seeking to compromise either the confidentiality or the integrity of the transmitted data.

4.1.2 IoT Malwares

BashLite

IoT DDoS malware which comes with a C&C [1]. Most BASHLIFE attacks are simple UDP, TCP floods and HTTP attacks. BASHLIFE infect a IoT device by brute-forcing its telnet access using known default credentials. One interesting aspect of BASHLIFE is that malware payload deployed in IoT devices has the BASHLIFE's C2s IP addresses hard-coded into it and are easier to monitor. Most of the infected devices are located in Taiwan, Brazil and Columbia. The source code of BASHLIFE was partly leaked in early 2015 and has led to many variants. BASHLIFE is considered the predecessor of Mirai and is in direct competition for vulnerable IoT real estate.

BASHLIFE/Lizkebab/Torlus/gafgyt is one of the popular malware which infects Linux based IoT devices to launch DDoS attacks. It was reported that BASHLIFE is responsible for enslaving over 1 million IoT devices, constituting mostly of Internet enabled cameras and DVRs. It is capable of launching attack of up to 400 Gpbs. Most BASHLIFE attacks are simple UDP, TCP floods and HTTP attacks. BASHLIFE infect a IoT device by brute-forcing its telnet access using known default credentials. One interesting aspect of BASHLIFE is that malware payload deployed in IoT devices has the BASHLIFE's C2s IP addresses hard-coded into it and are easier to monitor. Most of the infected devices are located in Taiwan, Brazil and Columbia. The source code of BASHLIFE was partly leaked in early 2015 and has led

to many variants. BASHLIFE is considered the predecessor of Mirai and is in direct competition for vulnerable IoT real estate.

BusyBotNet

This IoT malware comes with embedded tools usually found on full systems dedicated for ethical hacking. These toolset include Hydra, masscan, tshark and reverse shell backdoors.

Lightaidra

LightAidra/Aidra is a IRC-based mass scanning and exploitation tool support on several architectures, namely MIPS, MIPSSEL, ARM, PPC, x86/86-64 and SuperH. Malware is designed to search open telnet ports that could be accessed using known default credentials. The source code of LightAidra is freely available on the Internet as open source project. It allows scanning and exploiting routers for make BOTNET (in rx-bot style). In addition to this, with Aidra you can perform some attacks with tcp flood.

Linux.Wifatch

Another IoT malware which infects devices and enslave them in a C&C fashion.

Lizkebab

An IoT Botnet derived from BashLite.

Mirai

An IoT Botnet designed for DDoS attacks. It is known for being responsible for the Dyn cyberattack which occured in October 2016. It has a C&C malware, with an agent written in C and the controller developed in GO. Mirai is one of the most predominant DDoS IoT botnet in recent times. Mirai means "the future" in Japanese. Mirai botnet is definitely the next step in IoT DDoS malwares, however not as sophisticated as Remaiten but most effective. Mirai botnet is famous for being used in the record breaking 1.1Tbps DDoS attack with 148000 IoT devices. Mirai targets mostly CCTV cameras, DVRs, and hoem routers. Since the release of the Mirai source code, the number of IoT infected devices has increased from 213000 to 483000 in just two weeks. Mirai generates floods of GRE IP, GRE ETH, SYN and ACK, STOMP, DNS, UDP, or HTTP traffic against a target during a DDoS attack. More recently, Mirai has been found to be enhanced to infect Windows devices, helping hackers hijack even more devices. This enhanced Mirai could also identify and compromise database services like MySQL and Microsoft

SQL running on different ports to create new admin ”phpminds” with the password ”phpgodwith” allowing the hackers to steal the database. The awareness of IoT botnets in recent times attributes to Mirai and the volume of traffic generated during its DDoS attacks.

pnscan

This yet another IoT Botnet was designed as a parallel network scanner.

Randomware

An IoT ransomware.

4.2 Common IoT vulnerabilities & solutions

4.2.1 Introduction

IoT devices utilize a variety of standards and protocols for communications [19], whether it's for applications and messaging (XML, HTTP, CoAP ...), network and transport (DTLS, RPL, IPv6 ...) or physical communications (ZigBee, Bluetooth, WiFi, LoRa ...). This diversity makes it pretty challenging to design a robust generic security infrastructure, since the heterogeneous nature of IoT networks implies that we usually face hybrid networks. As a consequence, the more protocols and technologies we burden in our network, the larger the attack surface will be. Khan and Salah [19] carried out an in-deep study on how diverse these infrastructures can, while highlighting their strong interaction with Key Management Protocols, which is kind of the ultimate objective of this chapter.

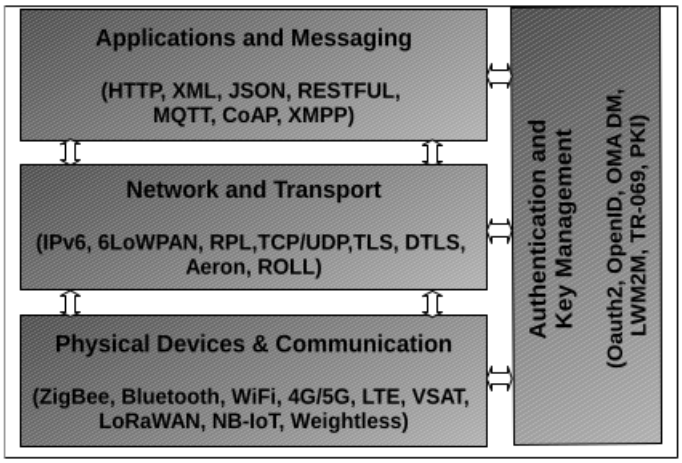


Figure 4.1: Source: [19] - Common IoT standards and protocols

As for every other connected device, IoT objects need to be uniquely identified on the network. For this purpose, nodes are usually identified by the mean of their MAC adress and IP adress. This mechanism leaves the network exposed to low-level Sybil and Spoofing attacks. This attack can be conducted by malicious Sybil nodes and may lead to serious resource scarcity, in an environment which is basically resource-constrained. Current object identification and locating mechanisms in IoT in general are exposed to a large number of vulnerabilities and operational attacks, making it a very active research topic [28].

The combination IPv6/Bluetooth is also as vulnerable as commonly used in WSNs. IEEE 802.15 standard requires fragmentation for IPv6 packets. Malicious nodes capacity to capture fragments and replay their duplicates may cause some security issues such as energy drainage or buffer overflows. This attack is known as fragmentation replay and duplication attack. IPv6 routing protocol is also susceptible to some attacks when combined with the Routing Protocol for Low-Power and Lossy Networks, known as RPL routing attacks. It's even more important to consider Ipv6 security, as Ipv4 won't be able to accommodate all connected devices on the long term with their exponential number increase.

These network standards are subject to several network attacks such as session establishment and resumption attack. This consists in forging messages in order to hijack a session. In this scenario, a malicious node can spoof another legitimate node's identity and act as a proxy between it and its counter part communicating node. In a word, weak security measures in the network will facilitate MITM-based attacks.

In addition to network related vulnerabilities, IoT objects can also present serious system issues, whether on the physical or the OS level.

Appendix B is a snippet on how vulnerable some of the commonly used network standards in Wireless Sensor Networks (WSNs) can be.

4.2.2 Real-context risks & mitigations

A fact which stands out when looking closer at the IoT security industry, is the huge gap between the adoption of IoT for business and the adoption of IoT security practices. We require IoT security by design to ensure lifecycle sustainability for IoT devices [2]. When looking at different IoT solutions developed in the industry, we first look at common issues and vulnerabilities as a starting nail to track corresponding solution products. OWASP IoT Top 10 [3] nicely summed up these vulnerabilities, giving an outer view of the threat landscape. In a addition to new and upcoming regulations, these issues make the conducting vectors driving the IoT security industry. No

wonder they are mostly technical based, since the very purpose of developing a security technology is to hunt some specific threat.

Default passwords

A first approach to solve the problem is to look at how this password hard-coding is realized, and so, work the process straight out. The first point is that these passwords are too weak. Using a wordlist of common IoT devices passwords, an attacker is able to carry out a dictionary-based attack and take over the device. So was the case of Mirai alike attacks as explained above. The second point concerns the use of that same password all along the devices life-cycle. Users never bother to change them, assuming they are fully aware that their device can be remotely accessed using that password. Third point is that these passwords are hard-coded, so they can be quickly retrieved with old fashion reverse engineering. Moreover, they are usually stored in plain text or in a weak hash scheme, such as MD5, in order to save computing power.

The first point can fixed by requiring secure default password policies, including password length, complexity and uniqueness. Second point could be solved using password expiration and one-time password. The latter will force users to change default passwords when setting up their IoT infrastructure. Once again, we assume here that users have been notified of their device using a password in the first place. The third point is abit more challenging to spear. As mentioned, it comes a consequence of the power-constraint nature of IoT devices. Hence, it's difficult to work it out, without wrecking the device's performance.

An alternative approach to address this issue is to sum up several authentication layers. Passwords is actually an authentication mechanism among others. Therefore, adding extra authentication can mitigate the risk of password compromise. This can be implemented through two-factor authentication, biometric authentication, Single Sign On tokens (SSO), digital certificates and PKI.

It's vital to bear in mind that this password issue is basically authentication related. Whereas authentication can be realized throughout who we are, what we know and what we possess. From this perspective, passwords is something we know. If it burdens vulnerabilities, we can just bypass it by exploring other information we know, alongside who we are and what we possess. An IoT devices fleet is usually relatively widespread geographically, making the latter two authentication strategies merely feasible.

Solutions

1. One-time passwords (OTP)

2. Temporary one-time passwords (TOTP)
3. Smart card based SSO
4. eSIM

Irregular patches and updates

Once the IoT infrastructure is set up, the device fleet remains mostly static, and that can lead to tremendous disasters. Devices come with embedded firmware. And like almost any firmware, it can be reverse engineered and pwned. Overtime, a firmware is susceptible to become obsolete and vulnerable. That's why patches and upgrades are vital against common security vulnerabilities and 0-Days exploits.

The Satori attack is a good example to illustrate the risk [4]. The attack occurred when hackers exploited critical vulnerabilities in wide range of routers (D-Link, Huawei, RealTek ...), allowing them to get a RCE, in order to build up large IoT botnets. These botnets were exploited at first to conduct massive DDoS attacks, and later on, to mine digital coins. Besides, Satori's payload include a worm to allow the infection of neighboring devices with no human interaction needed, in a Stuxnet fashion.

The problem with upgrading IoT devices resides in the fact that these are usually widespread out there in the field. It's difficult to deliver an update to a thermo-sensor deployed in a farm for instance. Some solutions exist though, even if they only solve the problem partially.

Solutions

1. Pre-signed URL

Insecure interfaces

IoT devices aren't standalone components. They implement network protocols for communication (e.g. Bluetooth, LPWAN ...), services for maintenance (e.g. Telnet) and applications for data processing and storage. These interfaces comes with their own vulnerabilities, whether generic or specific to the IoT implementation. Thus, they increase the attack surface.

The main strategy for mitigating this risk available right now is encryption. This particularly involves the use of digital certificates (e.g. X509). The ENISA report on "Good Practices for Security of IoT" [5] highlights above all security by design. Security has to be bared in mind all along the device's design and implementation, from the very cradle to the grave. The Best Practices to mitigate interface related risk involve application of ENISA good practices related to used standards. In other terms, one must apply security

measures not only limited to the device itself, but take a step further to pack in the entire ecosystem and bring it under control.

Solutions

1. Thales Trusted Key Manager
2. Remote subscription management
3. IQNOX IAM platform

Data exposure

Smart devices are specifically designed to carry out tasks where they handle large amount of data. A common risk in IoT security is data leakage. Scandals and large-scale attacks made lack of data protection in the IoT a top concern in the industry. Whether stored data at rest or active data being exchanged, it has to be safe from unauthorized access.

All communications at all layers have to be secured, how insignificant the layer might seem. An attack on a casino reported by Darktrace in 2017 demonstrates how expensive insufficient data protection can be [6]. The attackers in this case actually exploited lack of communications security in a smart thermometer of an aquarium exposed in a casino. They used this smart device to get a foothold in the casino's network, then going up to the cloud, and finally exfiltrate 10Gb of the casino's customers stored data.

The most efficient way at disposal to address this threat is cryptography. It defends against data theft for stored data and eavesdropping for over-the-network data. It preserves therefore the data privacy, integrity and confidentiality. Data at rest must, besides its encryption, be stored on an under-control host. When communicating, a secure channel with the counterpart shall be established and messages exchanged should be encrypted (confidentiality) and signed (integrity). Looking up to the previously described attack, we first notice the unencrypted communications which allowed the attackers to get their foothold in the first place. And second, the data were insecurely stored, which allowed the hackers to quickly fetch the database files and exfiltrate them.

Solutions

1. Thales embedded Secure Element (eSE)
2. Trusted Platform Module (TPM)

IoT fleet management

With the increase of connected IoT devices and the sophistication of its inter connection, the risk of misuse or mismanagement of some increases as well. A more and more common issue consist in IT teams not aware of what and how many connected devices they have inside their their network [7]. Within the same line, we are facing a rising number of "Shadow IoT and IoMT devices". Among the most dangerous discoveries, one concerns the use of Facebook and YouTube applications of medical devices operating Windows XP. Having a medical device connected running insecure web applications on an obsolete and unsupported OS is not only a cyber threat, but a threat to human life. This kind of mismanagement made such schemes a beloved target for ransomware attacks [8].

Therefore, it's crucial to deploy IoT management platform whenever it's necessary. IoT management platforms facilitates the monitoring of different devices connected to its network. It can be seen as an inventory of objects, which provides information about their status, connectivity, access logs, running apps and services ... etc. With such a global overview, one can set up security counter measures much easier, because this kind of information flow can easily fit in as an input for a SIEM or an IDS/IPS in order to automatically spot among others paranormal behavior, unusual access, collusion between subnets or VLANs, and foremost, shadow IoT devices. Obsolete or vulnerable devices can then be upgraded or patched. Improper network segmentation can be fixed. Shadow devices shall be disposed of.

Solutions

1. Azure IoT management platform
2. IBM IoT management platform

4.2.3 Legal framework evolution

Another vector driving the IoT security industry is the evolution of national and international regulations. Companies have to adapt their technical standards to meet requirements of laws which recently came into force. The EU voted the Cybersecurity Act in 2019, which already sets standards for IoT security manufacturers. The law was pushed by ENISA in order to curb IoT related risks. The US followed the trend with the IoT Cybersecurity Improvement Act late 2020. A bill passed on by British regulators require IoT devices sold in the UK to embed a "reasonable security feature".

Conclusion

The internship took place in two main parts, fundamental research and practical engineering. The first one was a furtherance the laboratory's research work on IoT security in general, and Group Key Management (GKM) in particular. During the second stage, I was rather wearing an engineer's hat. I was still working on IoT security, but more from a practical view.

The first part was by far longer and much more intense compared to the second one. This part itself went through different steps. The first step of my research task was dedicated to self-learning, documentation and foremost reading academic papers. Although I have a cybersecurity background, I started the internship with almost no prior academic knowledge about security in the IoT domain. I was even much less ranged when it comes to GKM related topics. Therefore, I had a lot to catch up for at first, in order to better apprehend different scientific advances and challenges. I had study as much as possible the concepts I was going to work on. This concerns different GKM schemes and the MGKMP at first. But then it extended to the Cluster Head (CH) architectures, when I started considering it for the developing solution.

The second step is an in-depth study of MGKMP. The primary purpose was to identify eventual contribution axes to the ongoing research effort. The forthcoming challenges identified included possible optimizations of the protocol's theoretical design and fixing some vulnerable aspects of the protocol's features. The ultimate goal has always been making a contribution to the existing work in order to produce a research article.

The third and final step of the first stage was to focus on a specific problem. The Key Manager's issue of single point failure was considered. A solution based on an election scheme was developed. Simulations showed some promessing results. A lot of work is yet to be done though, especially for the experimental part.

By the end of the first stage, that ultimate goal mentioned had been fulfilled. A paper summing up the developed solution was published in the proceedings of the *International Conference on Communications Software (SoftCOM 2021)*. This soluion was without a doubt crucial for the system. It solves the targeted problem and does what it is supposed to do. This has already been confirmed by international reviewers from the *SoftCOM 2021* conference. Nevertheless, I still have reserves regarding its utility and practical implementability. The work was carried out under a fundamental research program. But at some point, it has to be transposed into a practical engineering solution. Otherwise, the whole effort would be pointless. That's why I do believe this research should never be disconnected from the field realm. Permanent links and practical evaluations should be conducted all the way, to make sure that program doesn't loose its main obejective from its sight. The program's ultimate purpose is to defend an IoT network against threats. However, how good do we know this threat and how does it operate ? Will this system really deter a malicious hacker from attacking, even a little bit ? A field study of the attackers operational mode is crucial, to avoid defending the front door while the attacker is striking from behind. Foremost, its important to be confident that we are not switching to new security systems, which look safer from the outside, whereas it's rotten to the core from the inside. This said, it doesn't in any way systematically implies the system is useless. Because so much of the work achieved is brilliant and academics have confirmed it. But it's only to point some carency out, regarding the overall approach in the problem resolution.

I gained a lot of experience in this stage. It involved a tremendous professional research know-how. This includes how to conduct a bibliographic work, how to read academic papers and criticize them, how manage articles using academic softwares (*Zotero* in my case) and write my own article in LaTeX. The research requires a lot of precision, thinking and synthesis. One always has to take into account other researchers work and their outcome. The solution development require permanent focus on the ultimate goal, in a way that despite all the sideline details, one never gets diverted or distracted. The writing of the article and its publication gave me a closer insight of the academic domain.

Second part was focused on the IoT security industry. It consists mainly in studying common IoT security risks and industrial solutions from a practical point of view. Working on real IoT systems and malware samples was at most beneficial, as I gained useful knowledge and first hand experience on IoT engieering. It raised my awareness about several types of threats, and made me look closer at different engineering solutions.

I found the internship's overall unfolding very satisfactory on the personal level. I worked on a research task, and thus I have now a better aptitude

to work in the research field. I also had some engineering tasks, and thus I gained knowledge in case I want to work in the industry. This internship actually helped a lot in personal development and my professional plans. Besides, the pedagogic objective of an internship is to enhance my operability after graduating. This objective couldn't have been better fulfilled.

Appendix A

Simulation: Scenario n°1

```
[+] Starting Simulator v0.1 ...

[+] Simulation init ...
[+] Node init: ID=0
[!] CEF score for node 0 updated to: 222.33
[+] Node init: ID=1
[!] CEF score for node 1 updated to: 838.24
[+] Node init: ID=2
[!] CEF score for node 2 updated to: 402.56
[+] Node init: ID=3
[!] CEF score for node 3 updated to: 621.28
[+] Node init: ID=4
[!] CEF score for node 4 updated to: 716.45
[+] Node init: ID=5
[!] CEF score for node 5 updated to: 176.04
[+] Node init: ID=6
[!] CEF score for node 6 updated to: 16019.43
[+] Node init: ID=7
[!] CEF score for node 7 updated to: 1364.79
[+] Simulation initialized !

[+] Current group status ...

#### Node 1 ####
ID: 0
CEF Score: 222.33
Battery: 83.5 %

#### Node 2 ####
ID: 1
```

```
CEF Score: 838.24
Battery: 70.3 %

#### Node 3 ####
ID: 2
CEF Score: 402.56
Battery: 45.3 %

#### Node 4 ####
ID: 3
CEF Score: 621.28
Battery: 40.7 %

#### Node 5 ####
ID: 4
CEF Score: 716.45
Battery: 74.4 %

#### Node 6 ####
ID: 5
CEF Score: 176.04
Battery: 39.1 %

#### Node 7 ####
ID: 6
CEF Score: 16019.43
Battery: 78.4 %

#### Node 8 ####
ID: 7
CEF Score: 1364.79
Battery: 74.9 %

[+] Snapshot saved to snap00 !

[!] CEF score for node 0 updated to: 122.6
[!] CEF score for node 1 updated to: 734.24
[!] CEF score for node 2 updated to: 315.78
[!] CEF score for node 3 updated to: 528.52
[!] CEF score for node 4 updated to: 641.64
[!] CEF score for node 5 updated to: 153.16
[!] CEF score for node 6 updated to: 15085.57
[!] CEF score for node 7 updated to: 1244.71
[+] Node 1 running for elections ...
```

```
[+] Node 3 running for elections ...
[+] Node 4 running for elections ...
[+] Node 6 running for elections ...
[+] Node 7 running for elections ...
[+] Node 0 voted ...
[+] Node 1 voted ...
[+] Node 2 voted ...
[+] Node 3 voted ...
[+] Node 4 voted ...
[+] Node 5 voted ...
[+] Node 6 voted ...
[+] Node 7 voted ...
[+] Node 6 claiming KM role !
[+] KM role attributed to Node 6
[+] Node 7 claiming Supplicant role !
[+] Supplicant role attributed to Node 7

[+] Current group status ...

#### Key Manager ####
ID: 6
CEF Score: 15085.57
Battery: 75.9 %

#### Deputy KM ####
ID: 7
CEF Score: 1244.71
Battery: 52.4 %

#### Node 1 ####
ID: 0
CEF Score: 122.6
Battery: 65.1 %

#### Node 2 ####
ID: 1
CEF Score: 734.24
Battery: 48.7 %

#### Node 3 ####
ID: 2
CEF Score: 315.78
Battery: 24.6 %
```

```
#### Node 4 ####
ID: 3
CEF Score: 528.52
Battery: 21.5 %

#### Node 5 ####
ID: 4
CEF Score: 641.64
Battery: 52.8 %

#### Node 6 ####
ID: 5
CEF Score: 153.16
Battery: 23.0 %

[+] Snapshot saved to snap01 !

[!] CEF score for node 0 updated to: 94.51
[!] CEF score for node 1 updated to: 496.47
[!] CEF score for node 2 updated to: 159.03
[!] CEF score for node 3 updated to: 257.92
[!] CEF score for node 4 updated to: 445.88
[!] CEF score for node 5 updated to: 85.23
[!] CEF score for node 7 updated to: 851.92
[+] Node 1 running for elections ...
[+] Node 4 running for elections ...
[+] Node 7 running for elections ...
[+] Node 0 voted ...
[+] Node 1 voted ...
[+] Node 2 voted ...
[+] Node 3 voted ...
[+] Node 4 voted ...
[+] Node 5 voted ...
[+] Node 6 voted ...
[+] Node 7 voted ...
[+] Node 1 claiming Supplicant role !
[+] Supplicant role attributed to Node 1
[+] Node 7 claiming KM role !
[+] KM role attributed to Node 7

[+] Current group status ...

#### Key Manager ####
ID: 7
```

```
CEF Score: 851.92
Battery: 31.7 %

#### Deputy KM ####
ID: 1
CEF Score: 496.47
Battery: 28.0 %

#### Node 1 ####
ID: 0
CEF Score: 94.51
Battery: 48.3 %

#### Node 2 ####
ID: 2
CEF Score: 159.03
Battery: 5.7 %

#### Node 3 ####
ID: 3
CEF Score: 257.92
Battery: 4.7 %

#### Node 4 ####
ID: 4
CEF Score: 445.88
Battery: 33.0 %

#### Node 5 ####
ID: 5
CEF Score: 85.23
Battery: 8.3 %

#### Node 6 ####
ID: 6
CEF Score: 15085.57
Battery: 74.3 %

[+] Snapshot saved to snap02 !

[!] CEF score for node 0 updated to: 68.86
[!] CEF score for node 1 updated to: 268.6
[!] CEF score for node 2 updated to: 15.9
[!] CEF score for node 3 updated to: 21.14
```

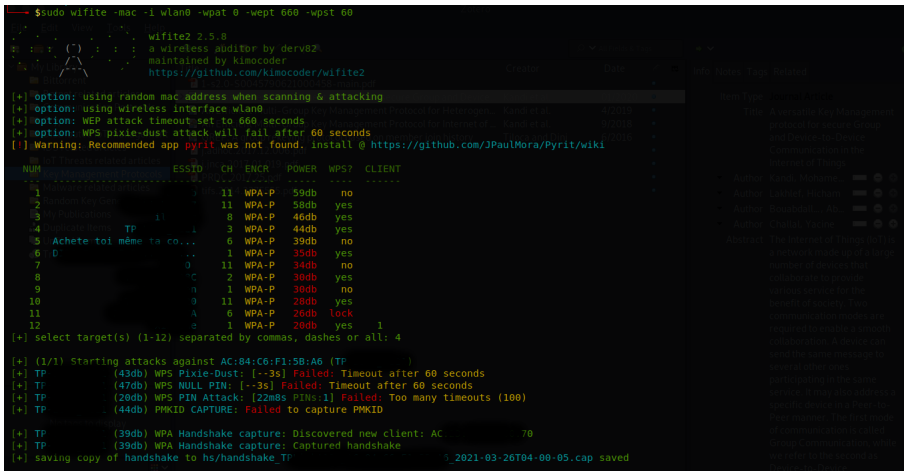
```
[!] CEF score for node 4 updated to: 266.44
[!] CEF score for node 5 updated to: 23.21
[!] CEF score for node 6 updated to: 14292.61
[+] Node 6 running for elections ...
[+] Node 0 voted ...
[+] Node 1 voted ...
[+] Node 2 voted ...
[+] Node 3 voted ...
[+] Node 4 voted ...
[+] Node 5 voted ...
[+] Node 6 voted ...
[+] Node 7 voted ...
[+] Node 6 claiming KM role !
[+] KM role attributed to Node 6

[+] Current group status ...

#### Key Manager ####
ID: 6
CEF Score: 14292.61
Battery: 72.2 %

#### Node 1 ####
ID: 0
CEF Score: 68.86
Battery: 33.1 %

#### Node 2 ####
ID: 1
CEF Score: 268.6
Battery: 10.9 %

#### Node 3 ####
ID: 2
CEF Score: 15.9
Battery: -11.4 %

#### Node 4 ####
ID: 3
CEF Score: 21.14
Battery: -10.5 %

#### Node 5 ####
ID: 4
```

```
CEF Score: 266.44
Battery: 15.9 %

#### Node 6 ####
ID: 5
CEF Score: 23.21
Battery: -5.0 %

#### Node 7 ####
ID: 7
CEF Score: 851.92
Battery: 19.1 %

[+] Snapshot saved to snap03 !
```

Appendix B

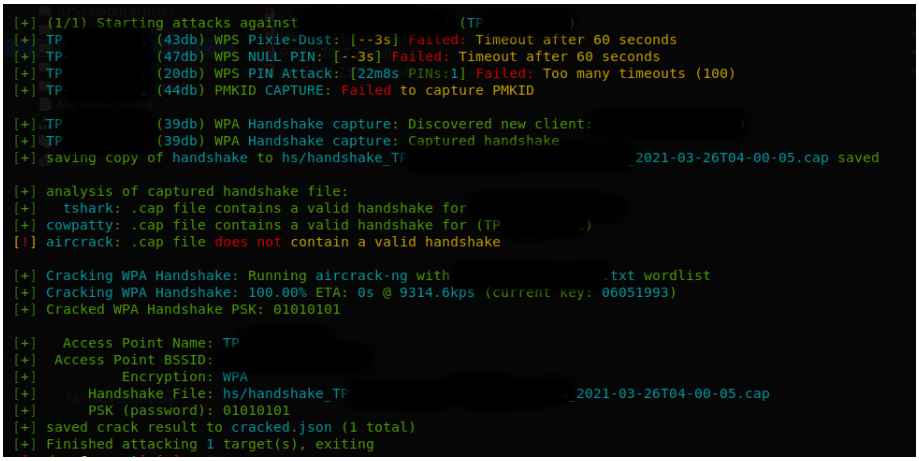
WPA2-PSK secured WiFi network attack

In the following example, we seek to compromise a WiFi encrypted network, since WiFi networks are commonly used for the transport layer in IoT architectures.



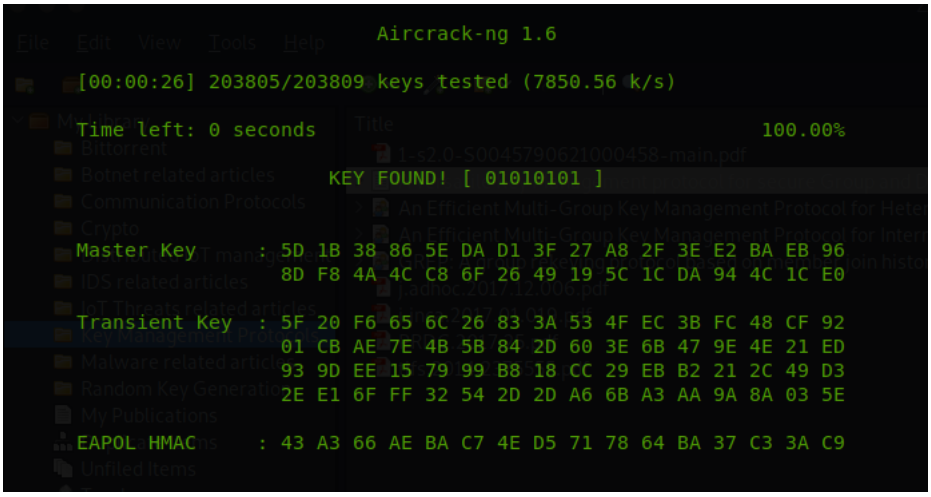
The first step is to technically identify the WiFi AP through ots ESSID and BSSID. The objective is to dump a handshake exchange with one client. If we can manage to identify a connected client to the network, we can try to cut it off from thr network by sending multiple deauth frames. This will force the client to reconnect to the AP, allowing us to capture the handshake. Once the handshake is captured, we crack it to obtain the WiFi network's key.

What we actually do is that we analyse the captured .cap file to extract the password. We use a dictionary-based attack in order to find a match for



the captured handshake by replaying the exchange for different possible passwords.

Using a dictionary-based password attack, we are able to recover the password in just few seconds.



Bibliography

[1] . https://securityintelligence.com/news/bashlite-malware-uses-iot-for-ddos-attacks/, 2021. [Online; accessed 13-July-2021].

[2] . https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/iot-security/key-principles, 2021. [Online; accessed 13-July-2021].

[3] . https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf, 2021. [Online; accessed 13-July-2021].

[4] . https://arstechnica.com/information-technology/2018/06/widely-used-d-link-modemrouter-under-mass-attack-by-potent-iot-botnet/, 2021. [Online; accessed 13-July-2021].

[5] . https://www.enisa.europa.eu/news/enisa-news/how-to-implement-security-by-design-for-iot, 2021. [Online; accessed 13-July-2021].

[6] . https://www.thesun.co.uk/tech/6062743/hackers-stole-casino-database-fish-tank/, 2021. [Online; accessed 13-July-2021].

[7] . https://www.helpnetsecurity.com/2020/07/24/analysis-of-5-million-unmanaged-iot-and-iomt-devices/, 2021. [Online; accessed 13-July-2021].

[8] . https://www.cpomagazine.com/cyber-security/rise-in-healthcare-data-breaches-driven-by-ransomware-attacks/, 2021. [Online; accessed 13-July-2021].

[9] Ahmed Al-Baz and Ayman El-Sayed. A new algorithm for cluster head selection in LEACH protocol for wireless sensor networks. *International Journal of Communication Systems*, 31(1):e3407, January 2018.

[10] Pieter Arntz. OVH cloud datacenter destroyed by fire. https://blog.malwarebytes.com/malwarebytes-news/2021/03/ovh-cloud-datacenter-destroyed-by-fire/, 2021. [Online; accessed 23-July-2021].

[11] Trupti Mayee Behera, Sushanta Kumar Mohapatra, Umesh Chandra Samal, Mohammad S. Khan, Mahmoud Daneshmand, and Amir H. Gandomi. Residual Energy-Based Cluster-Head Selection in WSNs for IoT Application. *IEEE Internet of Things Journal*, 6(3):5132–5139, June 2019.

[12] Maissa Dammak, Sidi-Mohammed Senouci, Mohamed Ayoub Messous, Mohamed Houcine Elhdhili, and Christophe Gransart. Decentralized Lightweight Group Key Management for Dynamic Access Control in IoT Environments. *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*, 17(3):16, 2020.

[13] A. B. Feroz Khan and G. Anandharaj. A cognitive key management technique for energy efficiency and scalability in securing the sensor nodes in the IoT environment: CKMT. *SN Applied Sciences*, 1(12):1575, December 2019.

[14] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. page 10, 2000.

[15] Dongyao Jia, Huaihua Zhu, Shengxiong Zou, and Po Hu. Dynamic Cluster Head Selection Method for Wireless Sensor Network. *IEEE Sensors Journal*, 16(8):2746–2754, April 2016.

[16] Mohamed Ali Kandi. *Lightweight Key Management Solutions for Heterogeneous IoT*. PhD thesis.

[17] Mohamed Ali Kandi, Hicham Lakhlef, Abdelmadjid Bouabdallah, and Yacine Challal. A versatile Key Management protocol for secure Group and Device-to-Device Communication in the Internet of Things. *Journal of Network and Computer Applications*, 150:102480, January 2020.

[18] Sang H. Kang and Thinh Nguyen. Distance Based Thresholds for Cluster Head Selection in Wireless Sensor Networks. *IEEE Communications Letters*, 16(9):1396–1399, September 2012.

[19] Minhaj Ahmad Khan and Khaled Salah. IoT security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, page 17, 2018.

[20] Djamel Eddine Kouicem, Youcef Imine, Abdelmadjid Bouabdallah AbdelmadjidBouabdallah, and Hicham Lakhlef. A Decentralized Blockchain-Based Trust Management Protocol for the Internet of Things. page 14, 2019.

[21] Yi-Hsuan Kung and Hsu-Chun Hsiao. GroupIt: Lightweight Group Key Management for Dynamic IoT Environments. *IEEE Internet of Things Journal*, 5(6):5155–5165, December 2018.

[22] Help Net Security. 41.6 billion IoT devices will be generating 79.4 zettabytes of data in 2025. https://www.helpnetsecurity.com/2019/06/21/connected-iot-devices-forecast/, 2019. [Online; accessed 31-May-2021].

[23] Marco Tiloca and Gianluca Dini. GREP: A group rekeying protocol based on member join history. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 326–333, Messina, Italy, June 2016. IEEE.

[24] Hien Thi Thu Truong, Miguel Almeida, Ghassan Karame, and Claudio Soriente. Towards Secure and Decentralized Sharing of IoT Data. page 8, 2019.

[25] I-Chen Tsai, Chia-Mu Yu, Haruo Yokota, and Sy-Yen Kuo. Key Management in Internet of Things via Kronecker Product. In *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 118–124, Christchurch, New Zealand, January 2017. IEEE.

[26] Luca Veltri, Simone Cirani, Stefano Busanelli, and Gianluigi Ferrari. A novel batch-based group key management protocol applied to the Internet of Things. *Ad Hoc Networks*, 11(8):2724–2737, November 2013.

[27] Furui Zhan, Nianmin Yao, Zhenguo Gao, and Guozhen Tan. A novel key generation method for wireless sensor networks based on system of equations. *Journal of Network and Computer Applications*, 82:114–127, March 2017.

[28] Zhi-Kai Zhang, Michael Cheng Yi Cho, Chia-Wei Wang, Chia-Wei Hsu, Chong-Kuan Chen, and Shiuhpyng Shieh. IoT Security: Ongoing Challenges and Research Opportunities. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pages 230–234, Matsue, Japan, November 2014. IEEE.

[29] Jan Henrik Ziegeldorf, Oscar Garcia Morchon, and Klaus Wehrle. Privacy in the Internet of Things: threats and challenges: Privacy in the Internet of Things: threats and challenges. *Security and Communication Networks*, 7(12):2728–2742, December 2014.