Reviewing and Optimizing the Security of FIGO

by

S.K.L. Chevalking

Commissioned by:

University of Twente, and
Twente Institute for Wireless and Mobile Communications

February 14, 2011

Thesis supervisors:

Dr. Zheng Gong, Prof. Dr. Ir. Pieter Hartel, Prof. Dr. Ir. Sonia Heemstra de Groot
and Dr. Ir. Hartmut Benz

Acknowledgements

I am most thankful to my supervisor Dr. Zheng Gong, for his constant support and rich feedback. Many thanks go out to Prof. Dr. Ir. Sonia Heemstra-de Groot and Dr. Ir. Hartmut Benz, who always were there for me to discuss my work and provide me with useful information. I am also thankful to Prof. Dr. Ir. P.H. Hartel and Dr. F.E. Kargl for their feedback.

I am grateful to Twente Intsitute of Wireless and Mobile Communications (TI-WMC) for giving me the opportunity to conduct my thesis at their company. I owe many thanks to the development team of TI-WMC and special thanks to Tom Lippman who has been a great help.

Finally, I would like to thank my wife and son for always believing in me. I would also like to thank my parents and parents-in-law for their support and encouragements.

Abstract

Public safety parties use wireless communications systems to enable emergency responders to communicate with each other as well as with the home department. Sensitive information is transported by emergency response systems. Therefore, emergency response systems need a strong security system to prevent information disclosure.

Key management is the cornerstone of every security system and weak key management can lead to a decrease in security and performance. The academic literature on key management in wireless networks is numerous and a sub-set focuses on emergency response systems.

In this thesis we research FIGO, a specific emergency response system that is developed for Dutch emergency responders. We show that key management in FIGO is sub-optimal and does not satisfy the security requirements for emergency response systems. We propose small changes to the network architecture to enable an improved key management scheme.

Our proposal is a key management scheme for emergency response systems: KERS. It aims to improve the security of FIGO and focuses on building up a secure ad hoc mesh network. KERS is built on the security requirements for emergency response systems. We theoretically claim that KERS improves the security of FIGO and does not incur performance penalties.

Contents

1. Introduction

Public safety parties are assigned the responsibility to prevent and protect the general public from incidents that can endanger them, like crimes and disasters. When such an event occurs, public safety parties must switch from their daily routine to an emergency program. The daily routine mainly consists of monitoring possibly dangerous situations. The emergency program consists of gathering, appointing and equipping groups to specific areas of the disaster, and briefing them of their tasks. The core public safety parties, or emergency response services, consists of police, fire rescue, and medical service.

During an incident, emergency response services need to communicate over short and long distances. Short distances involve for example, two police officers at the incident site being 300 meters apart. Long distance communication is considered communications with the back office, or home department. Short distance communications must be available without relying on existing communication infrastructures.

However, existing communication infrastructures might be unavailable during the time-span of the incident, to various causes such as overload or destruction. Long distance communication cannot function without an existing infrastructure like the cell-phone network or WiFi access points. Wireless communication technologies give numerous possibilities to provide communications for emergency response services during incidents.

1.1 Motivation

There is numerous literature on securing wireless ad hoc networks, and a subset focuses on public safety networks, or emergency response systems. Examples of such projects are MESA[38] and SafeCom[15]. In this thesis we study a commercial solution: the FIGO system.

FIGO is designed for Dutch public safety parties and offers connectivity for short and long-range communication. It builds up an independent communication infrastructure, and is able to use existing infrastructures to communicate over long distances. A FIGO network has three components. First, the *back office*, the physical department building of a public safety party, contains a back office node that connects to the mesh network. Second, the *mesh network* consists of FIGO mesh nodes that are mounted in vehicles. Third, *client devices* from emergency responders use the FIGO mesh as a communication infrastructure during incidents.

This thesis shows that the security in the FIGO mesh does not meet the standards for emergency response networks. Key management in the FIGO mesh is inadequate making the network vulnerable to attacks on data confidentiality, integrity and availability. This thesis aims to:

- research the strengths and weaknesses of the FIGO security architecture;
- discuss key management schemes that can be applied in the FIGO mesh;
- propose a key management scheme that improves the security of the FIGO mesh;
- show that the proposal can be applied to the FIGO system.

1.2 Organization

This thesis is organized as follows. The following chapter provides more background information on emergency response systems and presents a case study of the FIGO project. The case study includes a description of the FIGO project, and describes and reviews the security architecture of FIGO.

In Chapter 3, we discuss academic proposals of key management schemes that are suitable, or possess suitable characteristics, for emergency response systems. It also discusses the IEEE 802.11i standard, which is a widely used security protocol for wireless networks.

Chapter 4 introduces our solution to improve key management in FIGO. We briefly introduce our key management scheme for emergency response systems (KERS) by explaining the preliminaries and discuss the various states that our scheme has.

Chapter 5 presents the detailed scheme of KERS. The scheme is divided in different phases, and each phase is explained in detail. We show that our solution covers all situations that can occur during an incident, taking into account the mobility of the nodes.

In Chapter 6, we subject KERS to a theoretical security analysis to show that it does improve the security of FIGO. The scheme does not violate the security requirements for emergency response systems. Additionally, we provide a theoretical performance analysis to see how our solution affects the performance of the FIGO network. To conclude Chapter 6, we present two practical implementation issues of KERS.

Finally, Chapter 7 concludes the thesis and gives recommendations for future work. We take the future developments of FIGO into account and discuss how our scheme fits in the development of FIGO.

2. **Information Security Analysis of Emergency Response Systems**

The most critical communication problem for emergency responders during disaster relief operations is the inability to use existing communication infrastructures. Existing infrastructures can become unavailable due to destruction, collapse or overload. Radio communication is a good alternative for exchanging voice data, but emergency services require exchanging other types of data e.g. video or files. These data types give emergency responders a better overview of the situation and aid in a better handling of the incident.

Extending emergency response systems to support the exchange of video data and files introduces security risks. Video data and files may contain confidential data, which must not be disclosed to outsiders. Authentication, data integrity and availability are also important properties for emergency response systems.

Emergency response systems rely on wireless communications. Such networks resemble a mobile ad hoc network (MANET) or wireless mesh network (WMN). MANETs are flexible, self-organizing networks, which makes them suitable for environments that have mobile nodes. WMNs are self-configuring and self-healing, which is convenient in environments with instable connections, which wireless connections are by default.

In Section 2.1, emergency response systems are described from a security point of view. We specify security requirements for such systems, based on the literature and existing projects.

Section 2.2 introduces the main focus of this thesis, the FIGO system. We explain the architecture of the system, its functionalities and its applications as an emergency response system.

The case study is followed by a security analysis of FIGO in Section 2.3. The security analysis consists of describing the current security architecture of FIGO, which is followed by an evaluation of the security architecture.

The results of the evaluation are summarized in Section 2.4, followed by the conclusion that serves as input for the following chapters. The main conclusion of this chapter is that key management is an important function for an emergency response system, and the key management in FIGO is currently unacceptable.

2.1 Emergency Response Systems from a Security Perspective

An emergency response network consists of three components: A *back office* is the physical department building of the emergency service that functions as control center; an *ad hoc mesh network* consisting of mobile, mesh nodes; and *client devices* that enable emergency responders to communicate with each other and the back office. An emergency response system connects these three components.

Security measures in emergency response systems are preferably enforced by the back office, which has a central view of the network. When the back office cannot enforce security measures due to loss of connectivity, the mesh network must enforce security autonomously.

Emergency responders use client devices that connect to the mesh network in order to communicate with other emergency responders or the back office. Emergency responders can exchange data that is subject to security requirements.

2.1.1 Adversary Model

The adversary model specifies different types of adversaries that pose a threat to emergency response systems. Adversaries with the right equipment can eavesdrop on transmitted information within the emergency response network. If no security measures are taken, the attacker can read the information and thereby violating data confidentiality. The attacker is passive during this attack, and therefore called a *passive communication attacker* (*PCA*). A successful passive attack is a threat to data confidentiality.

When an attacker intercepts information and replays it in the network, the attacker is said to be active. The *active communication attacker* (*ACA*) is defined as an adversary who modifies communication flow in a network. The *ACA* can perform attacks resulting in unreadable messages. The *ACA* can inject self-made packets in the network, resulting in bogus traffic. Alternatively, an *ACA* can become an intermediate between two nodes to perform a man-in-the-middle (MitM) attack. By performing a MitM attack, an adversary controls and relays information on a link between two nodes. *ACA*s threaten authentication, authorization, data confidentiality, availability and integrity.

Adversaries who try to exhaust the network its resources are called *exhaustive attacker* (*EA*) attackers. Examples of such attacks are flooding the network with bogus messages or sending out strong signals, called signal jamming, to consume bandwidth. Deliberately consuming network bandwidth results in lower performance of the emergency response network or even denying legitimate use of the network.

Because mobile nodes in emergency response systems are mounted in vehicles, we stipulate that hardware attacks can be performed only by *inside attackers* (*IA*). Adversaries performing hardware attacks capture sensitive information directly from the node. The *IA* can read key material from node memory, which threatens data confidentiality and integrity. *IA*s can also tamper with the node, such that it controls the node. This threatens authentication and authorization.

Table 1: Adversaries and their capabilities

Adversary/Attacker	Capability	Acronym
Passive communication	Sniff communications in the network	PCA
Active communication	Perform replay attacks, man-in-the-middle attacks or modify the communication in the network	ACA
Exhaustive	Flood the network, or deliberately consume the network's resources	EA
Inside	Capture information from the node, tamper with the node	IA

2.1.2 Security Assumptions

Any emergency response system acts in a specific environment with certain assumptions. Our security assumptions specify situations in which the security architecture can be taken for granted.

First, we assume that mobile nodes are physically attached to a vehicle's dashboard, and cannot be captured by adversaries. Adversaries can only gain physical access to a mobile node by taking the emergency vehicle away from the incident site. Emergency personnel are present during incidents, which makes it infeasible to tamper with a mobile node during an incident. As a second consequence, mobile nodes are assumed to have sufficient power, because they do not rely on small batteries. Instead, mobile nodes make use of the car battery/generator to provide them with power.

Our second assumption is made due to a lack of time to complete this thesis. We assume that an emergency response system as described in this section only applies to the technical security solution. A technical solution must always be designed based on

the human policies and capabilities to see if it is a realistic solution that can work in practice. We realize that this is a flaw in our design, but time restrictions force us to make this assumption.

Finally, we assume that mobile devices are able to communicate securely in their emergency service center with a trusted server that is connected to the back office node. When nodes receive credentials, adversaries have no chance of intercepting them. The security assumptions can be summarized as follows:

- **Assumption 1 (A1):** Node capturing by adversaries is restricted.
- **Assumption 2 (A2):** Mobile nodes have a sufficient power supply.
- **Assumption 3 (A3):** Human policies with regard to security are assumed to be trustworthy.
- **Assumption 4 (A4):** Emergency response systems rely on a secure configuration phase, invulnerable for attacks.

Security concepts for emergency response systems can be derived from theoretical proposals and practical implementations of such systems. The most important security concepts are device and channel authentication, authorization, confidentiality, efficiency, integrity, key management, non-repudiation, message authentication, scalability and service availability [15][23][37][40].

Authentication refers to proving that an entity's claim of identity is true. Secure authentication protects against impersonators. Authentication comes in two types: device authentication, which is needed to access the network, and channel authentication that ensures a channel between two entities is authentic and not shared by others. *Authorization* processes, or access control, decides which entities are given what permissions and privileges in the network. *Confidentiality* means that information is accessible only to those who are authorized to access the information. *Efficiency* refers to the computational complexity, energy consumption and communication overhead of the security solution. The concept of *integrity* means that the message

contents are not altered intentionally or maliciously during transmission without being detected. *Key management* includes generation, storage and distribution of key material that is used in secure communications. *Non-repudiation* holds when the sender of a message cannot deny having sent the message, and when the receiver of the message cannot deny its reception. *Message authentication* provides a means to verify the integrity of the message and possibly verify the sender. *Scalability* means that the security solution should scale well, when the number of nodes and number/size of messages in the network increases. *Service availability* refers to the self-healing capabilities of the network, when it is faced with a disruption of communication channels.

The security concepts lead to the following security requirements for emergency response systems. When possible we have based the requirements on literature, when references are absent the requirements come from FIGO or our own experience.

- **Authentication (R1 and R2)**
    - All participants in the emergency response system must prove their identity to other participants. This includes the back office node, mobile client devices and mobile nodes.
    - Bi-directional communication channels between two parties must only be accessible by the two parties.
- **Authorization (R3 and R4)**
    - Communication between nodes of emergency response services must be able to communicate only within that network. Nodes are not allowed to communicate with nodes from other emergency response services.
    - Only authorized mobile client devices must be allowed to access the emergency response network.
- **Confidentiality (R5)**
    - Confidential data must be encrypted to prevent passive attackers from reading it.
- **Efficiency (R6 and R7)**

  - Communication overhead for security must be minimized to optimize network performance.
  - The back office must be able to exercise control over different sub-groups of the network.

- **Integrity (R8 and R9)**
  - The integrity of important data must be ensured to prevent active attackers from inserting, modifying or copying data into the network, without being detected.
  - The system must use message origin authentication to ensure data integrity and detect inserted messages by active adversaries.
- **Key management (R10 and R11)**
  - The system must support over the air re-keying (OTAR), allowing key material to be updated over the network they protect or another secure channel.
  - The system must use separate keys for confidentiality, integrity and authentication, to minimize the impact of a key compromise.
- **Non-repudiation (R12)**
  - Nodes must not be able to deny having sent or received messages
- **Scalability (R13)**
  - Security mechanisms must be scalable to support network and member dynamics, since it is not known in advance how many emergency vehicles will be present at an incident site and now node mobility will affect the network
- **Service availability (R14 and R15)**
  - Security in the emergency response system must be self-healing when faced with exhaustive attackers.
  - Security in the system must be automatically restored after a key or device compromise.

This section described the basic functionality of emergency response systems, and specified an adversary model, security requirements and assumptions for such systems. To build up these properties, a well-designed key management scheme is necessary. Next, based on this security architecture, we introduce the FIGO system. We can

evaluate FIGO by matching the security criteria for emergency response systems with the security of FIGO.

2.2 FIGO: A Case Study

The goal of FIGO is to provide emergency response services with a communication infrastructure. FIGO builds its infrastructure on multiple wireless communication technologies like, IEEE 802.11 [26], UMTS/HSDPA and satellite communications (SatCom).

FIGO consists of mobile physical routers, referred to as FIGO nodes or mobile nodes. Mobile nodes are mounted in vehicles, attached to the vehicle's power supply. FIGO is a multi-hop network that enables long-distance communication, as long as there are enough nodes between the sender and receiver to forward the message. The network model of the FIGO network is presented in Figure 1.

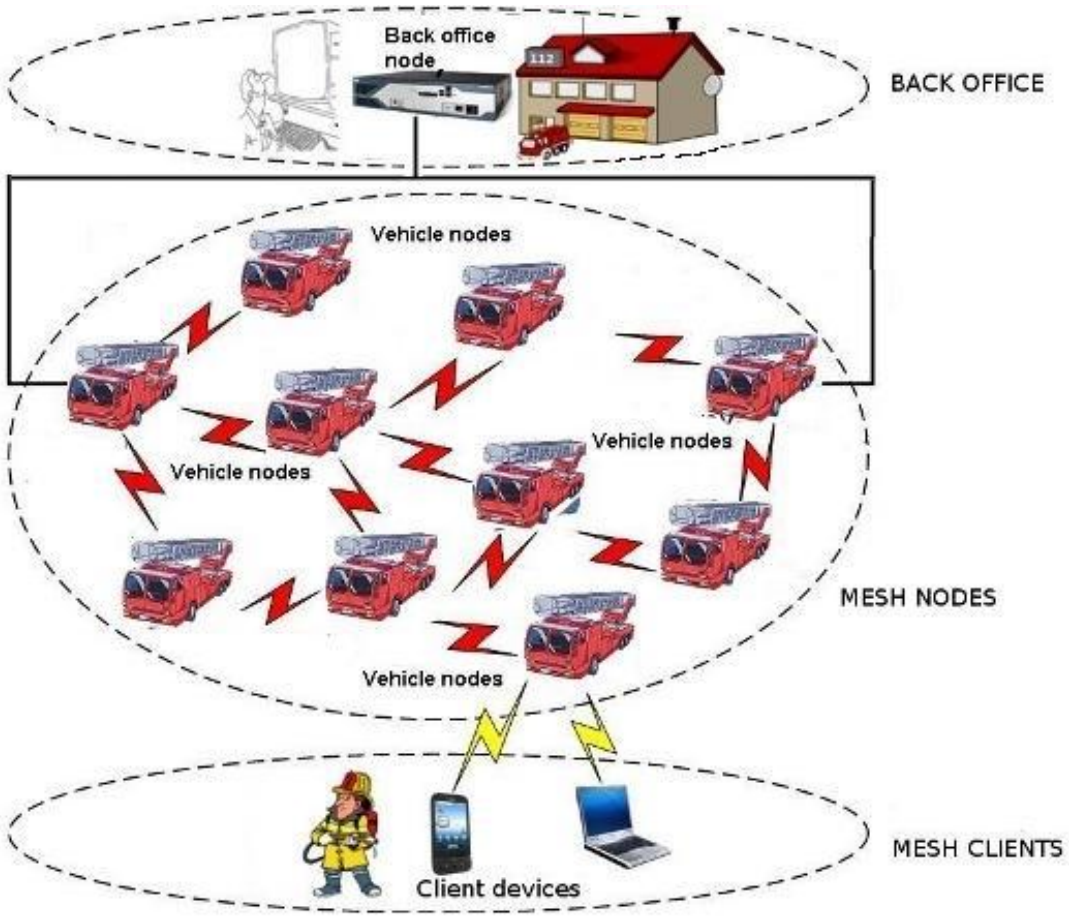

Figure 1: FIGO network model

The self-configuring capability of the FIGO nodes gives the network its ad hoc nature. When mobile FIGO nodes come in each other's vicinity, they connect to form a communication network, called the FIGO mesh. Mobile FIGO nodes are capable of storing and forwarding information, and maintaining a routing table. The FIGO mesh is the core of the network infrastructure.

When a public communication infrastructure is available, mobile FIGO nodes can set up a connection with the back office containing a FIGO back office node. The connection is made over the Internet and enables the back office to get an overview of the incident and communicate with emergency responders. The back office node has an overview of the FIGO mesh and can update credentials to ensure the security of the network.

Emergency responders carry mobile client devices, e.g. laptops and PDAs that can connect to the FIGO network to enable the exchange of information. Mobile client devices can also connect to the back office and receive information about the incident, using the FIGO mesh.

To get a better understanding of the security of the mobile FIGO nodes, we discuss the protocol stack of the mobile node. The protocol stack consists of different layers, each responsible for part of the communication. Figure 2 illustrates the protocol stack and each layer of the protocol is described in more detail.

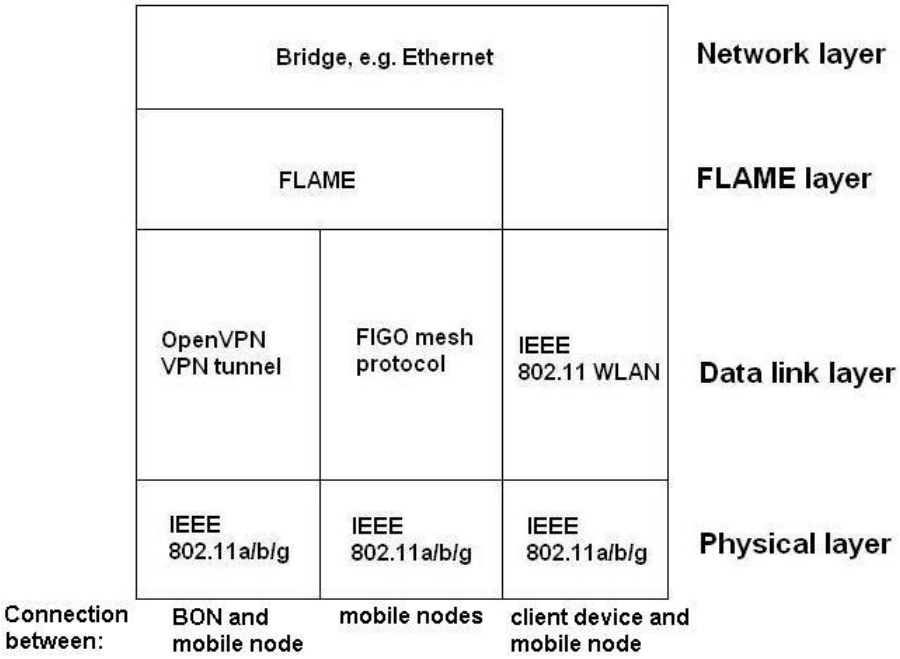

Figure 2: FIGO protocol stack

Physical layer: The physical layer in the FIGO mesh consists of one of the IEEE 802.11 protocols, e.g. 802.11a/b/g. The physical layer in FIGO deals with the transmission of raw bits of information over the air. Security in physical layer is not present.

VPN tunnel: A virtual private network (VPN) is used as a secure channel to connect geographically remote places through a public network, usually the Internet. FIGO uses site-to-site VPN to connect the mobile nodes with the back office.

VPN uses tunneling protocols as opposed to layered protocols. Tunneling protocols encapsulate data packets within the private network and transmits them through the Internet. There are two main reasons to encapsulate data packets, the first is to make the data packet compatible with the public network, enabling the use of standard transmission protocols. The second reason is to enhance security of the data packet.

VPN ensures authentication, confidentiality and message integrity. Security mechanisms in VPN are independent of security mechanisms in the public network. A data packet can be encrypted in VPN, and encapsulated in the public network.

FIGO does not use VPN in the FIGO mesh, because secure broadcast communication is not supported in VPN and it slows down the connection speed. Unstable connections cause mobile nodes that use VPN to re-establish the connection often, which degrades the network performance.

Mesh layer and AP/LAN layer: The data link layer of FIGO consists of the mesh layer and access point (AP)/local area network (LAN) layer. The data link layer is responsible for the data transfer between entities in the LAN, the mobile nodes and mobile client devices. The mesh layer is responsible for data transfer in the FIGO mesh, while the AP/LAN is responsible for the data transfer between the mobile node and mobile client device. In FIGO, security mechanisms are not implemented at the data link layer, but in layers above.

FLAME: The forwarding layer for meshing (FLAME) enables multi-hop communication, which is not supported by IEEE 802.11/a/b/g. FLAME resides between the media access (MAC) layer and the network layer (e.g. IPv4/IPv6).

Transmission requests from the network layer are handled by FLAME before they proceed to the MAC layer. FLAME inspects the destination link in the address field and determines how the message should be sent over the MAC layer. FLAME uses its routing table to determine the next hop to the destination address. The destination address can be anywhere in the network. Without FLAME, the destination address always would have to be a direct neighbor of the sender.

The advantage of FLAME is that it works regardless of the implementation of the network and MAC layer. Neither needs to be changed. FLAME does not include any security mechanisms that provide data confidentiality or integrity. In FIGO nodes, security mechanisms are added to the FLAME layer to ensure data confidentiality and integrity.

From the network model and protocol stack, it shows that security in FIGO concentrates on multiple entry and exit points. Such points are defined as points where information enters and exits the network. Data flows from mobile client devices to the FIGO mesh, from the FIGO mesh to the back office and from mobile FIGO nodes to other mobile FIGO nodes. The data flows are bidirectional, which makes each exit point also an entry point.

2.4 Information Security Analysis of FIGO

To assess the security of FIGO we assume that the adversary model, security assumptions and requirements of Section 2.1 apply for FIGO. Based on this security model we describe the security mechanisms of FIGO.

Following the different entry and exit points, the security mechanisms of FIGO can be divided in three parts. First, we describe how security is implemented to protect the data flow between the FIGO mesh and the back office. Then, we describe the security mechanisms to protect the data flow between mobile client devices and the FIGO mesh. Finally, we describe which security measures are taken to protect the FIGO mesh.

Requirements for the connection between the back office and the FIGO mesh can be derived from the types of data that are sent. The channel between back office node (BON) and a mobile FIGO node (MN) must not be shared with other entities. This can be ensured by mutual authentication. The back office and FIGO mesh exchange confidential data when mobile client devices communicate with the back office. The back office and the FIGO mesh exchange control traffic to maintain an optimal performance in the FIGO mesh, i.e. routing updates. It is important that the integrity of the control traffic is protected, meaning that fake or erroneous control messages must immediately be detected.

The connection between BON and MN is secured with VPN technology. FIGO uses OpenVPN [46] as software solution to create a VPN connection between BON and MN. Before a VPN connection is established, the MN and BON authenticate each other by verifying their certificates, providing mutual authentication. OpenVPN uses OpenSSL for data encryption and supports HMAC to ensure data integrity. OpenVPN provides security mechanisms for authentication, confidentiality and integrity.

Mobile client devices use the FIGO mesh to transfer user data that is highly confidential and requires message integrity. Only mobile client devices that belong to emergency personnel is allowed to access the FIGO mesh. This requires access control and authentication.

FIGO implements a secure solution to secure the connection between client devices and the FIGO mesh. Access to the FIGO mesh is secured by the WPA [45] or IEEE 802.11i [26] standard. Both protocols provide credentials for data confidentiality and integrity.

Although authentication can be provided by the standards, FIGO takes a different approach that is less time consuming and does not require a trusted third party. Instead of authenticating the mobile client device with an external server, the mobile FIGO node authenticates the client device based on its device ID and the possession of the

corresponding key. The key that is used in these protocols is pre-shared, meaning it is pre-distributed to the mobile client device before it connects to the network. The key is transferred in plaintext on a USB stick to the mobile client device.

Nodes in the FIGO mesh forward all the above mentioned data types and require the above mentioned security requirements. *Authentication* in FIGO is performed by open system authentication (OSA) [26], which is based on the SSID of the wireless access point. OSA has been criticized for its lack of security [3] and its weak authentication mechanism.

Data integrity in FIGO is preserved by using a cyclic redundancy check (CRC). CRC is an error-detection code and can be used to detect non-accidental data alterations. A CRC code *c* is computed over message *m* and attached to the message. The sender sends message *mc* to the destination node. When the node receives *mc* it computes *c'* over *m*, using the same function as the sender. If *c'=c* then message *m* has not been altered. Otherwise an alteration in message *m* has been detected and the receiver can discard the message.

Confidentiality is ensured using 128-bit AES encryption. The key is generated as a random 128-bit string and is transferred to the nodes during a key pre-distribution phase. Every node that belongs to the same FIGO network, receives the same key. The lifetime the key is of arbitrary length. An AES hardware module takes care of the encryption and decryption.

The pre-distribution stage to transfer the credentials requires confidentiality and authentication. Distribution of all credentials to the mobile FIGO nodes is done using a secure configuration channel. During the configuration phase, wireless communications are disabled. Configuration can only take place with a physical connection, e.g. Ethernet or USB connector. The connection creates a secure channel, assuming that the mobile node is in the same room as (or in sight of) the system administrator.

This section presented the security mechanisms that are implemented in FIGO. Assuming that the security mechanisms function properly, data confidentiality, integrity, and device authentication are ensured. However, the security mechanism can still have flaws that break the security requirements. An evaluation of the security mechanisms will carefully review each mechanism and how it satisfies the security requirements, and which security requirements are not met in FIGO.

2.5 Evaluating FIGO Security

When evaluating FIGO security mechanisms we first look at how the security mechanisms cover the security requirements for the data flow. This is a straightforward process, since it has been described in Section 2.4. Then we evaluate if the other security requirements from Section 2.1 are covered.

OpenVPN is an open-source technology for VPN-tunneling. The standard provides data confidentiality, integrity and authentication [46]. Certificate-based authentication ensures mutual authentication before the VPN tunnel is established.

The security mechanisms that are used to restrict mobile client devices to access the FIGO mesh, e.g. WPA and WPA2/IEEE 802.11i provide authentication, data confidentiality and integrity, and access control [26].

Data confidentiality in the FIGO mesh is ensured by AES-128, which is heavily scrutinized and a widely accepted standard to protect data confidentiality.

Data integrity is protected with CRC, which is not accepted as a secure mechanism for data integrity. A CRC only detects data alterations, but does not provide message authentication and non-repudiation. A CRC can be created independently, without any knowledge of network secrets, thus adaptive communication attackers can alter a message containing a CRC – including the CRC itself.

Authentication is performed by using a combination of open system authentication (OSA)[26] and implicit authentication by the possession of the right encryption key.

OSA has been criticized as a method of authentication, and has been replaced by 802.1x [27].

OSA also serves as a form of access control since it separates different networks. The back office controls the assignment of SSIDs and determines which nodes are allowed to participate in which networks. SSID as access control mechanism only works if the authority to change SSIDs is exclusively preserved for the back office.

From the above mechanisms, we conclude that the security requirements regarding authentication (R1), access control (R3 and R4), confidentiality (R5), and efficiency (R6 and R7) are met on all three connection types. Because in the FIGO mesh nodes do not share pairwise keys, node-to-node unicast communication is not supported. Theoretically, it is possible to use the public key from the certificate for unicast communications, but this is not done. This leads to R2 not being met on channels in the FIGO mesh. From R8 to R14, we briefly examine each requirement to evaluate whether and on which connection type it is met.

The integrity requirements R8 and R9 are met with OpenVPN and WPA/IEEE 802.11i [26][45][46]. In the FIGO mesh these requirements are not met, because CRC does not provide the desired level of security. A CRC is easily forgeable; any participant can change the message and re-compute the CRC, resulting in the violation of R8. CRC is not bound to an identity resulting in a lack of message origin authentication and violating R9.

Only key management in OpenVPN meets the requirements of emergency response systems. OpenVPN updates keys dynamically and uses separate keys for confidentiality, integrity and authentication. Over the air re-keying is infeasible for both mobile client devices mobile nodes, violating R10. Additionally, for mobile nodes R11 is violated because separate mechanisms use a single key or no key at all.

Only OpenVPN supports non-repudiation in FIGO. Since both other mechanism use a single pre-shared key, a message is not bound to a single sender based on the

encryption key. Message reception cannot be proven, because message signatures are not unique.

Except for the OpenVPN solution, the other security mechanisms suffer from scalability issues. Although VPN connections are more expensive in terms of financial and bandwidth costs, in FIGO the number of VPN connections is minimized by not creating a OpenVPN connection from each node to the back office. For the FIGO mesh and mobile client devices, it holds that regardless of the increasing number of nodes in the network, the same encryption key is used. When more traffic is encrypted with the same key, the security of the key decreases. Thus, all mechanisms in FIGO scale badly in terms of performance, but only WPA/802.11i and the solution in the FIGO mesh decrease the level of security.

With regard to service availability, only OpenVPN meets the security requirements. OpenVPN connections will automatically be restored, but require re-authentication. When a node is compromised while there is a connection with the back office, the compromise will be detected and the node is placed on a black list (R14). Since keys are unique per link, a key compromise does not affect other links (R15). In the FIGO mesh and with the mobile client devices one service availability requirement is not met. After a successful attack from an exhaustive attacker communication can resume without re-authentication, thus R14 is met. Because of the pre-shared key that cannot be updated over the air, R15 is violated.

Table 2: Summary of the relationship between the connection type and the security requirements

Connection	**Satisfies requirements**	**Does not satisfy requirements**
BON ↔ MN	R1 through R15	-
Client device ↔ MN	R1 through R9, R11, R14	R10, R12, R13, R15
MN ↔ MN	R1, R3 through R7, R14	R2, R8 through R13, R15

From Table 2 we conclude that the connection between the BON and MN satisfies the security requirements for emergency response systems. We state that the performance penalty as a result from a successful attack performed by an exhaustive attacker is acceptable. The second conclusion we draw from our security analysis is that the static, pre-shared key for mobile client devices leads to not meeting three out of the fifteen security requirements. Since IEEE 802.11i also supports dynamic key updates, we consider the use of a pre-shared key a design choice for FIGO. Improving the security for mobile client devices falls out of the scope of this thesis. We advise to use follow IEEE criteria drafted in IEEE 802.11i.

The security mechanism in the FIGO mesh fails to meet half of the security requirements. Most of them are related to the fact that over the air re-keying (OTAR) is not supported. Others are because of the lack of secure data integrity and the lack of separate keys for separate security concepts. We conclude by stating that in order to satisfy the security requirements the key management scheme in the FIGO mesh must be redesigned.

This section has given an overview of the strengths and weaknesses of the security in the FIGO system. After concluding that security mechanisms OpenVPN and WPA/IEEE 802.11i are suitable security solutions for FIGO, we continue to improve the security in the FIGO mesh.

2.6 Conclusion

As a conclusion of this chapter, we summarize the security weaknesses of the FIGO mesh. By presenting abuse cases that illustrate the weaknesses, we justify our choice to redesign the key management scheme for the FIGO mesh.

The abuse cases concentrate on the adversary model of Section 2.1. From the security requirements, we state that passive communication attackers cannot violate the security in the FIGO mesh because confidentiality holds. Passive attackers can abuse bandwidth when mobile client devices of those attackers also use IEEE 802.11a/b/g to communicate. We consider this as a threat to performance, not to security.

Because confidentiality is ensured in the FIGO mesh, active communication attackers (ACA) cannot decrypt communications. However, the lack of a secure integrity-preserving mechanism gives ACAs the opportunity to replay packets into the network. Such replay attacks are a threat to security, enabling man-in-the-middle (MitM) attacks. An adversary relays all communication from node *A* to node *B* and vice versa. Adversaries can modify encrypted data to make it unreadable for the receiver. However, when the CRC is correct, the receiver will try to decrypt the invalid packet, while it could be dropped immediately with a standard intergrity-preserving mechanism. This results in the least in a performance threat.

When an administrator manually configures a mobile node, errors can occur that result in the node being unable to communicate in the FIGO mesh. When emergency responders connect their mobile client device to a FIGO node using the Ethernet port, they gain a direct access to the network, without authentication. When an adversary can capture an emergency response vehicle, he acquires a full access to the node. The attack can only take place when the node is offline, otherwise it would be noticed and emergency response personnel can apprehend the attacker. Consequently, abuse of direct access threatens data confidentiality, integrity and availability.

3. Key Management Schemes Related to Emergency Response Systems

The security analysis of FIGO has shown that the key management scheme of FIGO does not meet the security requirements of emergency response systems. The main drawbacks are the inability to provide over-the-air re-keying and not all security mechanisms use separate keys. In addition, FIGO has a weak data integrity protection, based on CRC instead of widely accepted standards. Academic proposals for key management schemes in similar systems provide us with characteristics for a secure scheme for emergency response systems. This knowledge helps us to design a key management scheme that improves the current key management in FIGO.

The function of a key management scheme in emergency response systems is to define the protocols for key generation, distribution, and usage. Key management also specifies how to securely store and update keys. The importance of node mobility in emergency response systems gives node addition and revocation a prominent role in key management.

There are challenges in designing a secure key management scheme for emergency response systems. The first is to update key material while minimizing the chance of network partitioning, which splits the network in two components that are unable to communicate with each other. The second challenge is to handle node mobility while minimizing the message overhead, which degrades the network performance.

This chapter discusses five proposals that focus on different network properties. The proposals are divided in two categories that are based on the hierarchical network model of the proposal. Section 3.1 presents proposals that have a *non-hierarchical network model*, where all nodes in the network have the same role and responsibilities. SMOCK [23] is a memory efficient scheme designed for emergency response systems and uses a public key infrastructure (PKI) where each node possesses a key ring. SOMA [8] is a self-organized scheme that uses certificated-based authentication. The scheme also uses a PKI and handles node addition and revocation in an efficient

manner. Both protocols do not rely on a trusted third party (TTP), and are completely autonomous and have no hierarchy.

Section 3.2 presents proposals that do have a *hierarchical network model*. Certain nodes are assigned a special role with responsibilities related to key management that distinguishes them from other nodes in the network. BALADE [6] is a scheme that works with group and local controllers. The network is divided into clusters, with the possibility to give each cluster its own symmetric group key. CLIQUES [42] also introduces group controllers to the network and the scheme uses a PKI, based on Diffie-Hellman [16] to generate and distribute a group key. The last protocol in this section is the widely used and accepted standard IEEE 802.11i. The protocol introduces a TTP that handles authentication and distribution of the credentials used for secure communication.

The schemes in this section are first briefly explained in terms of the network model and key properties. The network model describes how the entities and their relationships in the key management scheme. Under key properties, we describe which keys are used and what their function is.

Then the key management schemes are evaluated in terms of scalability, mobility support and member dynamics. *Scalability* measures if the network performance and security of the key management scheme degrades when more nodes make use of the scheme. *Mobility support* measures the extent to which unstable connections and mobile nodes cause overhead that degrades network performance and security. *Member dynamics* refer to the impact that node addition and revocation have on the network performance.

The reason why we only evaluate schemes on these three measures is twofold. First, these concepts are missing or inefficiently implemented in the current FIGO key management scheme, while the security requirements are essential in emergency response systems. Second, we consider that experts reviewed the security of the

discussed schemes, and their approval is shown by the acceptance of the work in academic journals.

In Section 3.3, the advantages and disadvantages of the protocols are discussed and evaluated. Based on this conclusion we present characteristics that make a secure key management scheme for emergency response systems, in particular FIGO.

3.1 Non-Hierarchical Approaches

In non-hierarchical approaches, every node has the same functionality and responsibilities, with regard to key management. Non-hierarchical approaches emphasize the autonomous, self-organizing character of ad-hoc networks.

This section presents two different non-hierarchical schemes to show the advantages as well as well as disadvantages for such an approach in emergency response systems. We discuss schemes that are related to mission-critical applications, although not always such an example could be found.

SMOCK [23] is a key management scheme designed for mission-critical applications. It focuses on emergency response services in incident areas, and the use of wireless ad-hoc networks during such situations. SMOCK uses a public key infrastructure and relies on an offline, trusted third party (TTP). The TTP is only needed during the pre-distribution of credentials and is therefore not considered a part of the network. Otherwise, SMOCK would have a hierarchical approach.

A group of nodes is connected when being registered to a single TTP, which manages a key pool containing a set of private-public key pairs. Each node holds a subset of private and public keys, which are distributed during a pre-distribution phase. When N_i sends a message to N_j, N_j has to know the set of public keys from N_i, where the corresponding set private keys only belongs to N_i. Keys are local, meaning that the revocation of a single key does only affect a sub-group of nodes.

In their proposal, the authors illustrate the protocol with an example including 10 nodes and 5 distinct key pairs. Each node holds 5 public keys and 2 private keys. This adds up to 7 keys. To ensure that only one node can decrypt secret communication, the message has to be encrypted multiple times. In the example, messages are encrypted two times, because every node has a unique combination of two private keys. Additionally, SMOCK requires the exchange of identities, before encrypted messages can be sent. Based on the received ID the sender can look up which encryption keys must be used.

When the size of the network is unknown beforehand, key pre-distribution can cause scalability issues. Distributing more keys than necessary will cause inefficient memory management, while distributing fewer keys can cause a shortage in keys. However, it must be noted that distributing private keys is not a problem; however, a shortage of public keys can occur. SMOCK provides a way to handle dynamic network sizes. New nodes must broadcast newly generated public keys in the network. The keys need to be signed by the TTP, so they can be verified.

SMOCK has a small memory footprint, a public keys can support $\Theta(2^a/\sqrt{a})$. The number of keys is $O(log(n))$, where n is the total of nodes. In traditional public key management schemes this is $O(n)$. SMOCK is resistant to the Sybil attack [18], where an adversary claims multiple rogue nodes to break the security of the scheme. The scheme also fulfills the data integrity, authentication, data confidentiality, non-repudiation and service availability requirements.

The scalability of SMOCK depends on the number of pre-distributed keys. If this number is too low. a shortage of keys will occur and if this number is too high, it will cause inefficient memory management. Node addition when there is a key shortage causes the joining node to broadcast newly generated public keys and previous deployed node must verify, register and store these keys. This is considered a fair

amount of overhead in control messages, which can become unacceptable when many nodes perform such a join operation.

Due to the lack of a network hierarchy, node mobility is supported. However, nodes can only move between networks that possess keys from the same TTP, which gives problems when different emergency services temporarily want to communicate with each other.

Member dynamics in SMOCK is supported. Node addition does not affect other members when keys are pre-distributed. Neighboring nodes only need to add the node to their table, together with the corresponding keys. This requires one authentication message. Because messages are encrypted multiple times, node revocation has a weak impact. If N_i, which possesses private keys 3 and 5, is compromised, nodes that possess key 3 or 5 can still function, because they have a different second private key. The compromise of multiple nodes leads to a bigger problem. If nodes are able to communicate misbehavior to peers and notify them of a revocation operation, nodes will have to delete the corresponding public and private keys. Private keys can be generated locally, but public keys need to be signed and updated by the TTP. Since the TTP is not online, this cannot occur dynamically. In the best case, a node joins the network with the signed public keys. As long as the node has not arrived at the incident site, secure communication cannot be guaranteed. In conclusion, member dynamics are supported in case of node addition, but lead to poor performance in case of node compromise.

SMOCK cannot provide dynamic key updates, because this requires an online TTP. Additionally, the scheme relies fully on public key cryptography, which requires certain hardware specifications that might not be affordable for emergency services. The scheme is memory efficient and satisfies most of the security requirements for emergency response systems. However, the lack of node mobility support, absence of a

revocation mechanism, and lack of dynamic key updates make it unsuitable for emergency response systems.

SOMA [8] is a self-organized key management scheme that creates a large-scale authentication system, without the need of a TTP. It uses certificate-based authentication in multi-hop ad-hoc networks. SOMA is built on top of Chord [44], which provides a scalable look-up algorithm that is robust on member dynamics and serves as a stabilization protocol.

Each node in SOMA creates its own public/private key pair and produces a self-signed certificate. When a node is deployed, it contacts its direct neighbors. A side channel (physical or any other direct channel) serves as an offline CA. Public keys of two peers are signed over the side channel. The public key is added to the certificate chain. As a result, all one-hop relationships are bi-directionally verified and direct neighbors serve as trust anchors. Node addition and certification work on principles of the Chord system using the certificate chain and trust anchors. Revocation of nodes can be implicit or explicit. The former takes place when a certificate expires, while the latter occurs when a node becomes aware that its private key is compromised. The node will explicitly revoke its public key certificate by using a revocation certificate, which will separate the relationship between the node and its ID. SOMA uses Chord to distribute the revocation request in an efficient manner, avoiding that revocation affects all nodes in the network.

SOMA is a secure scheme that prevents man in the middle attacks, impersonators attacks, and denial-of-service attacks. The scheme provides mutual authentication, data integrity, confidentiality, and message origin authentication. Additionally it does not use a TTP or CA, which emphasizes and enforces the autonomous character of the ad hoc network.

The scheme supports node mobility, because there is no hierarchy and when nodes move, only a neighbor discovery must be performed to adjust the Chord look-up table. Finding a chain of certificates in SOMA will not take longer than $O(log\ N)$ in time and number of certificates for any given trust path, which shows the scheme is efficient.

Due to the Chord system, node addition and revocation are supported without performance penalties. Key updates can be enforced locally and only affect nodes that are directly linked to the updated node, based on Chord. This avoids the transfer of sensitive key material and minimizes the chance for key material interception by adversaries.

Some aspects make SOMA less suitable for an emergency response system. The scheme does not scale well when multiple nodes join the network in a short period. When this occurs, much workload is placed on updating the Chord data.

The absence of a TTP is not always desirable in emergency response systems. For some operations, a form of centralized control is needed. In SOMA, this is difficult to enforce and results in a performance penalty, although the impact of that penalty is unknown.

SOMA serves as a good basis for the autonomous part of an emergency response system, if it is extended with a module that enables the centralized control, it will be a suitable key management scheme for emergency response systems.

3.2 Hierarchical Approaches

Hierarchical approaches enforce the structure in a network topology by allocating roles and responsibilities to a sub-group of nodes. A common role within hierarchical approaches is that of the group controller. Group controllers are responsible for key distribution within a sub-group, or cluster, of the network.

In this section, we discuss four key management schemes that make use of a network hierarchy. The schemes show the use of a network hierarchy in emergency response systems, as well as the disadvantages.

BALADE [6] is designed for mobile ad hoc networks (MANETs) and focuses on the mobility of nodes, while optimizing energy and bandwidth consumption. The scheme is designed for resource constraint applications, like military and public safety systems. BALADE is an adaption of the DEP protocol [17], which uses a single group key for secure multicast communications.

A hierarchy divides the network into different clusters. Contrary to clustering in DEP, which is static and does not support node mobility, BALADE introduces dynamic clustering. This adaption to DEP results in the support of node mobility, which makes the scheme suitable for military and public safety applications. The network hierarchy states that one group controller (*GC*) manages one cluster and is responsible for key generation and distribution. The *GC* is not considered part of the network, i.e. the *GC* is not a mobile node in a emergency vehicle. A cluster consists of group members (*GM*s) that are authorized to join the network. BALADE introduces local controllers (*LC*s), which are responsible that a cluster of nodes receives the group key. An *LC* is considered to be part of the network. Each *LC* has a list of its local members, which contains multi-hop destinations and not only its neighbors. Group members must obtain permission to become a *LC*, which is given by the *GC*.

BALADE requires multiple key encryption keys (KEKs), one for each cluster, to securely transfer the group key. The scheme does not have additional costs for encryption and decryption operations within a cluster. The network hierarchy has a positive impact on member dynamics, because node addition only affects the cluster and not the whole network.

BALADE uses a dynamic clustering scheme to optimize energy consumption and latency for key delivery, but the network must support a global positioning system (GPS) in order to use the clustering scheme.

BALADE is a secure key management scheme, but is too simplistic for complex operations. The scheme assumes there is always a connection with the *GC*. Additionally, all clusters use the same group key, and compromising a key affects the whole network. Node revocation must be communicated to all the *LC*s, which results in message overhead.

The scheme scales well because of the clustering scheme and supports node mobility. The concept of BALADE forms a good basis for key management in emergency response systems, but it requires a more efficient revocation mechanism.

CLIQUES [42] explicitly mentions the advantages in key management when having clusters in a network. The authors state that group controllers are necessary for managing group membership, making node addition and revocation more efficient. CLIQUES makes use of the Diffie-Hellman (DH) [16] protocol. CLIQUES shows that two-party DH can be extended to group DH (GDH) that can be used in dynamic groups. The security of GDH is proven in [43] and is based on the theorem that if a two party DH key cannot be distinguished from a random value, an *n*-party DH key cannot be distinguished from a random value.

The scheme distinguishes two different types of operations, the first relate to initial key agreement (IKA) and the second to auxiliary key agreement (AKA). IKA requires each participant to contribute a share to the group key. IKA has two stages, the first lasting (*n-1*) rounds, *n* being the total number of participants, and the second is a one round stage. In the first stage, each participant contributes its share to the group key. The second stage lasts one round and comprises of distributing the (*n-1*) intermediate values. Based on this value, each node identifies its intermediate value and computes the final group key.

When a node joins an existing group, it becomes the new group controller (*GC*). The *GC* saves the last message from the first stage of the IKA and extends the protocol with one message, when a new node wants to join the group. The new node performs

the second stage of IKA and the new group key is computed. As an alternative, the scheme also prescribes a fixed group controller, which suits stable environments without mobile nodes better, since there is no chance that the *GC* is unavailable.

When multiple nodes join a group, the node-addition protocol is inefficient, which is the main reason that CLIQUES specifies a mass join operation. Multiple nodes perform the IKA in such a way, that the second stage only has to be performed once. Only one new key is computed, while multiple nodes have joined the group.

CLIQUES specifies a group fusion operation, which differs from a mass join with respect to the pre-existing relationships in between nodes in both groups. The protocol leaves room for future work as it does specify multiple proposals of how to handle a group fusion. CLIQUES suggest that the smallest group joins the larger group similarly as a mass join, or that both groups use their key residues to compute a new group key. The security requirements of the application should determine which solution is appropriate.

Only the *GC* can revoke nodes. When revocation is successful, the *GC* computes a new group key, changing its own private key, and computes a new broadcast message that is sent to the group members. The group members can compute the new group key without having to change their private key. Because the revoked node does not receive this broadcast message, it cannot deduce the new group key. Only when the *GC* is compromised, a new *GC* has to be appointed and the IKA protocol has to re-run completely.

CLIQUES is a secure protocol for group-key agreement. The authors state that their scheme needs to be extended with member authentication and that group fusion needs to be further specified. Additionally, CLIQUES does not address the issue of scalability and states that increasing group size makes the protocol expensive. In addition, key integrity is not guaranteed, leaving the issues around authenticated key agreement.

The scheme serves as a basis for key management schemes that support node mobility and dynamic key updates. The use of Diffie-Hellman in CLIQUES is a useful idea since it allows node to establish secure channels without a bootstrap phase. If the scheme is extended to cover the aforementioned weaknesses, it is suitable for emergency response systems.

IEEE 802.11i (or 802.11i) [26] is the IEEE standard for wireless local area networks (WLANs). It is the successor of the Wireless Equivalency Protocol (WEP) [25], which flaws were pointed out by various literature e.g. [2][3][5]. In 2004, the IEEE ratified the successor of WEP, IEEE 802.11i or WPA2. The industry, in name of the Wi-Fi Alliance, had bridged the gap between WEP and WPA2 with their own solution Wi-Fi Protected Access (WPA) [45].

802.11i uses the term Robust Security Mechanism Association (RSNA) to refer to the strongest security algorithm currently defined in the standard. Other security algorithms are Pre-RSNA, and supported for backwards compatibility. RSNA uses two data confidentiality protocols, the Temporal Key Integrity Protocol (TKIP) and the Counter-mode/CBC-MAC (CCMP) protocol. RSNA includes an 802.1X [27] authentication and key management protocol.

The RSNA establishment procedure replaces the insecure Open System Authentication and Shared Key Authentication from WEP. The procedure can be divided into six stages as is done in [22]. RSNA provides strong mutual authentication and generates fresh traffic encryption keys (TEKs) for the data confidentiality protocols.

802.11i divides entities into three different categories. The node that wants to authenticate itself to the network is called the Supplicant (SUP). A SUP authenticates to a node or access point that is already part of the network, called the Authenticator (AUTH). The third entity is the Authentication Server (AS) (e.g. hosting a RADIUS

server [39]) that is used to perform authentication. 802.11i states that AUTH and AS roles can be performed by the same entity.

A successful authentication requires SUP and AUTH to mutually authenticate to each other and generate a shared secret that is used for deriving future keys. RSNA offers port-based access control through 802.1X, which ensures that no other data than authentication data can be exchanged during the authentication phase. If authentication is successful, the port is opened and SUP can participate in network communications.

802.11i uses the Extensible Authentication Protocol (EAP) [41] as an authentication framework. Using EAP, AUTH is able to forward messages from SUP to AS.

Stage 1: The first stage consists of the discovery of an AUTH with the desired security capabilities. The AUTH broadcasts RSN information elements (RSN IEs) in the plain, containing the security capabilities. AUTH can also reply to a Probe Request, sent by the SUP. Meaning SUP can participate actively and passively in access point discovery.

Stage 2: Upon successful neighbor discovery, authentication and association must be initiated. SUP sends a message to AUTH containing its security capabilities. At this stage, both entities are aware if an RSNA can be established. Stage 2 establishes weak authentication and the 802.1X port remains blocked such that no data packets can be exchanged.

Stage 3: This stage handles strong, mutual authentication. EAP-TLS [41] over 802.1X is considered the strongest possible protocol of mutual authentication. AUTH relays this communication to AS, if they are separate entities. The RADIUS server verifies the authentication information, and as a result, SUP and AUTH are mutually authenticated and share a Master Session Key (MSK). The SUP derives a Pairwise Master Key (PMK) from the MSK. Then AS and AUTH securely exchange the MSK and AUTH can derive PMK.

Stage 4: This stage consists of the 4-Way Handshake between SUP and AUTH that must be executed for a successful RSNA establishment. The handshake serves multiple purposes. First, it ensures the existence of the PMK and verifies the selection of the cipher suite. A cipher suite specifies the algorithms used for authentication, encryption and message authentication codes (MACs). The handshake also serves to derive a fresh Pairwise Transient Key (PTK) that is used in the data session for unicast communication. In addition, the handshake can be used by AUTH to distribute the Group Transient Key (GTK) for broadcast communication. After a successful handshake both AUTH and SUP share a PTK, optionally a GTK, and the 802.1X port is opened for data packets.

Stage 5: This stage is only of significance if the GTK is not exchanged in stage 4. During this stage, the AUTH generates and exchanges the GTK to the SUPs. This stage may be repeated using the same PMK.

Stage 6: The final stage is characterized by secure data communication using the PTK, for unicast, or GTK, for multicast/broadcast communication.

Although the protocol is secure, different attacks can be launched like the security level rollback attack and the reflection attack [22]. Besides the practical threats these attack may cause, there is another disadvantage when using 802.11i. As the protocol shows, it exchanges a lot of information before Stage 6 is reached. In addition, since this information is relayed in the most secure option, it is send twice. Intuitively key management in 802.11i is a time-expensive matter. This might not be problematic when nodes join the network and communication streams have not started.

802.11i does not support node mobility requiring re-authentication and re-association and the protocol lacks a protocol that specifies dynamic key updates, although the preliminaries are available after the four stages of key agreement are completed.

Although 802.11i imposes a network hierarchy on the nodes, the network is not divided in clusters. Still, node addition does not affect the whole network, only AUTH, but node revocation leads to a key update for all nodes. Member dynamics are thus not fully supported.

The scalability of the standard is dependent on the number of nodes that are assigned the role of AUTH and the number of AS. However, it is conceivable that those entities can form a single point of failure. As a result, the scheme its scalability is up for debate.

802.11i is a well-respected standard in wireless environments, but the scheme is too time expensive for emergency response systems, and does not support node mobility. The protocol is not designed for autonomous ad hoc networks, and must be extended to be suitable for emergency response systems.

3.3 Comparison of the Different Approaches

Based on the discussed protocols we can draw conclusions about the different approaches of key management, regarding the use of a network hierarchy. This section provides a comparison of both approaches. We compare the approaches on important characteristics for emergency response systems.

Scalability: Scalability is of importance, because during an incident the number of emergency vehicles varies per incident and is unpredictable to some extent. We can state that the number of vehicles will not exceed the available vehicles from a department. This changes when network interoperability is considered. The scalability of a key management scheme refers to the extent that performance remains stable when the number of emergency vehicles/mobile nodes approximates the total number of emergency vehicles/nodes that are available for the emergency service department.

Node mobility: Node mobility is a characteristic, because emergency vehicles are free to move during incidents. Incident areas can be large and nodes move from one part of the network to another.

Member dynamics: In literature, member dynamics is also referred to as the "1 affects *n*" phenomenon [6][14][31]. It refers to the extent to which a change in the network affects all the other *n* nodes in the network. Node addition and node revocation should have a minimal impact on the other nodes. Due to node mobility, this characteristic is of high importance.

Non-hierarchical approaches using a single, symmetric key score well regarding node mobility. Since nodes do not differ from each other, nodes can move freely without complex re-authentication mechanisms and acquiring a new key. Hierarchical approaches that use a different key for each group do not score well. A node moving between groups, need to re-authenticate and acquire the corresponding key to continue communications.

Hierarchical approaches are advantageous when scalability is an issue and the dynamics of joining and revoked members occur frequently. Revocation in emergency response systems is handled by the back office, which implies the presence of a hierarchy. When a connection with a group controller is possible, the advantages of the hierarchical approach regarding member dynamics and scalability benefit the emergency response network.

Non-hierarchical approaches that use a single symmetric key do not score well regarding member dynamics. When a node is compromised, all *n* nodes in the network need a key update. Hierarchical approaches that divide the network in different groups score better on member dynamics, if the groups use a different group key. When all groups use the same key, the effect is the same as with a non-hierarchical approach.

Table 3: Overview of key management schemes and their advantages and disadvantages

Key management scheme	Advantages	Disadvantages
SMOCK	+ Memory efficient + Scalable + Node mobility + Node addition does not affect the whole network	- Large amount of control traffic - Over the air re-keying not supported - Node revocation does affect all nodes
SOMA	+ Node mobility + Node addition and revocation do not affect the whole network + Minimal message overhead	- Not scalable
BALADE	+ Scalable + Node mobility	- Builds on unrealistic assumptions - Node revocation affects the whole network
CLIQUES	+ Supports over the air re-keying + Node mobility + Establish secure channels, without a bootstrap phase	- Scales badly - Node addition and revocation are sometimes inefficient - No authenticated key agreement
IEEE 802.11i	+ Scrutinized and widely used standard + Supports over the air re-keying	- Node mobility creates a large overhead - Node revocation affects the whole network

4. Introduction to KERS: A Practical Key Management Scheme for FIGO

Before we present our solution for a key management scheme for emergency response systems (KERS), we introduce the foundations on which KERS is built. We have made some adaptations to the current network model of FIGO, based on the findings in Chapter 3. KERS combines different characteristics from existing key management schemes to satisfy the security requirements.

The security model of KERS is equal to the security model presented in Section 2.1. We only make a distinction between data requirements and key requirements, meaning that data confidentiality is a requirement for FIGO while key confidentiality is a requirement for KERS. Because this is the only substantial difference between the requirements, we do not repeat the security requirements for KERS in this section, but refer to Section 2.1.

In Section 4.1, an adapted network model is presented, based on the network model of FIGO and findings form Chapter 3. The network model of FIGO relies on a single entity for key distribution. The disadvantages of this model can be mitigated by adapting the network model.

Next, in Section 4.2 we give a brief overview of KERS. In this overview, we describe the various phases of KERS and their relationships. The detailed descriptions of each phase give an insight how the scheme can improve the security of FIGO and its applicability in emergency response systems.

Section 4.3 gives a detailed overview on the keys that are used in KERS. Additionally, it describes the messages that are sent in KERS and what their compositions are.

This chapter connects the information gathered from academic literature and practical solutions to our solution, detailed in Chapter 5. We show that the security of FIGO can be improved without making changes to the system architecture.

4.1 Adapted Network Model for FIGO

The original network model of FIGO includes three different entities: back office nodes (BONs), mobile nodes (MNs) mounted in emergency response vehicles, and mobile client devices. The first shortcoming of this network model is that only the BON can distribute the group key. This introduces a single point of failure. Moreover, when the BON is unable to perform its task, performance is degraded because entities will not receive key material to encrypt data communications.

Chapter 3 pointed out several advantages of having a hierarchy in the ad-hoc mesh . One of the advantages is that key distribution can be divided to multiple entities. This section presents an adapted network model based on [7] that introduces and defines group controllers and local controllers. The extension to the network model is made to support a key distribution model that does not depend on one entity, but divides the task of key distribution over multiple entities. When the workload of key distribution is divided between multiple entities, the problem of a single point of failure is mitigated.

The original entities in FIGO remain unchanged, but their roles are extended to fit a distributed key management scheme. The certificate authority is added to the network model. Figure 3 shows the adapted network model. It shows that one BON can manage multiple networks, each controlled by one group controller (*GC*). The figure also illustrates that each network managed by a *GC* must have at least one local controller (*LC*). All other nodes are group members (*GM*). An independent FIGO mesh is not managed by a *GC* and contains only *GM*s. The functionality of the *GC*, *LC*, and *GM* will be explained in this chapter.

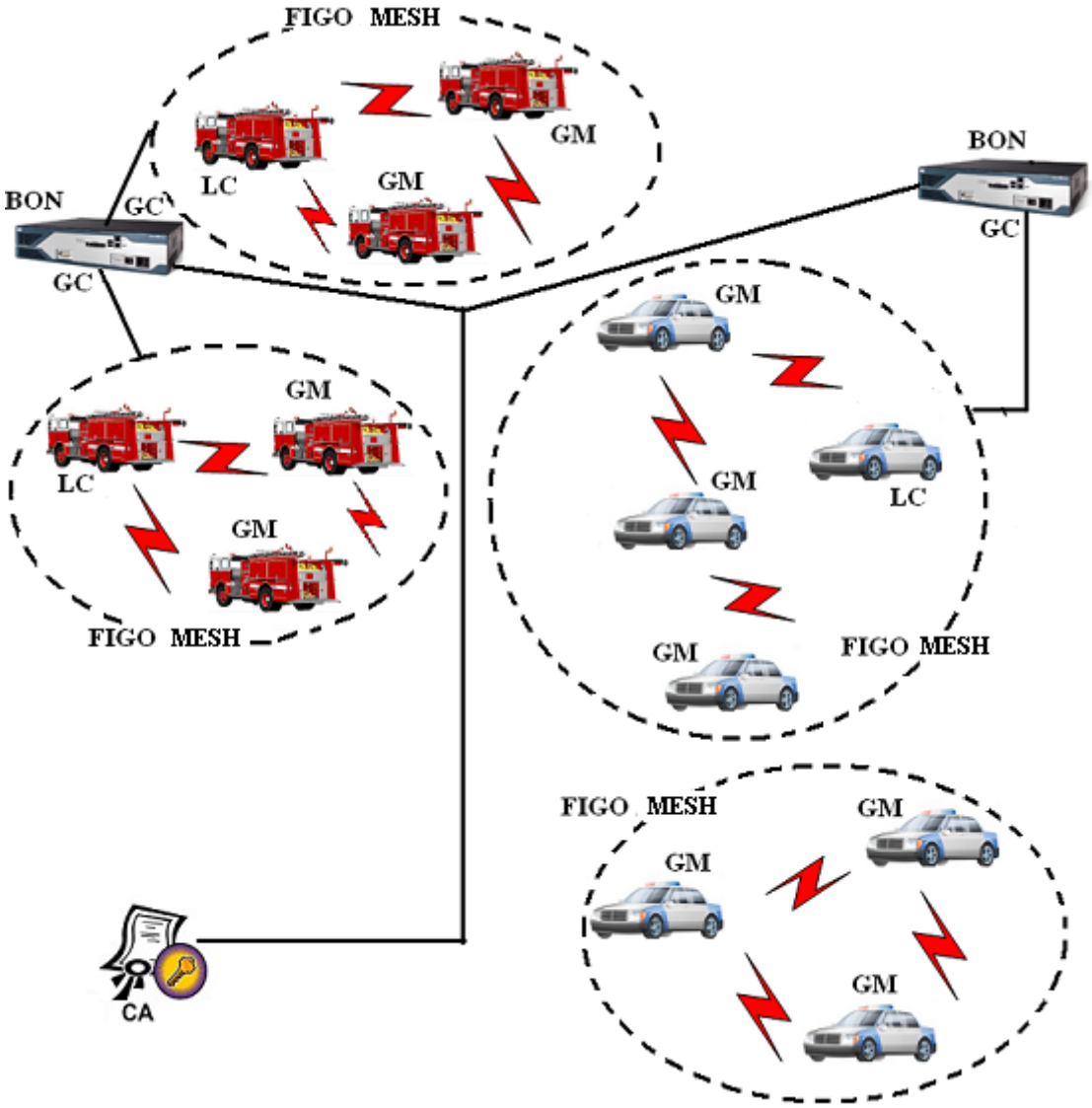

Figure 3: Adapted network model

Certificate authority: Device authentication in FIGO is performed implicitly by proving the possession of the current group key. In Chapter 2, the weaknesses of this method is illustrated. To improve device authentication in FIGO, nodes prove their legitimacy by verifying their certificates. The certificate is signed by a certificate authority (CA), which can decide to give BON the authority to sign certificates in name of the CA. Additionally when nodes are authorized to communicate with each other, but belong to a different BON, a certificate chain is used to perform authentication on a mutual shared trusted party. The following figure shows a possible certificate chain for FIGO.

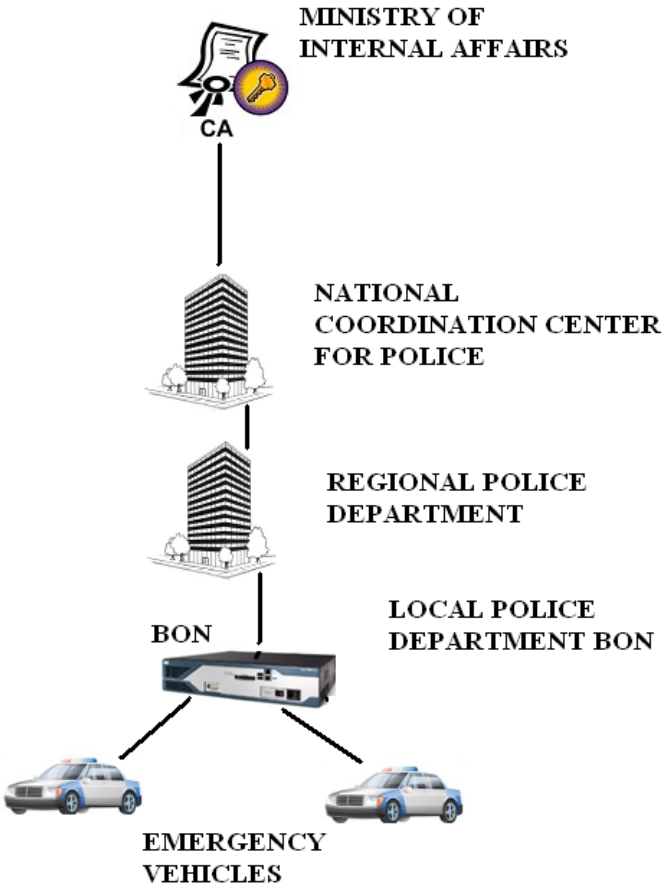

Figure 4: An example of a certificate chain in FIGO

Group controller *(GC)*: The concept of a group controller is not new in FIGO. BON currently serves as a single group controller. However, developments in the specifications of BON have lead to a redefined concept of group controller. As a result, the physical BON can manage multiple FIGO mesh networks by having multiple network addresses. Each network address is managed by a group controller (*GC*), which is a virtual entity on the BON that manages one FIGO mesh network.

The responsibility of a *GC* is to create a secure broadcast domain. The *GC* registers nodes that belong to the same FIGO mesh network. When an incident occurs, emergency vehicles assemble at the incident site and the mobile nodes will try to connect with the back office. If it succeeds, the *GC* will provide the network with a group credential that serves as parameter for the group key. The *GC* is responsible for updating the group credential before it expires. Another responsibility of the *GC* is to maintain a list of nodes that are authorized for communication. The list also contains nodes that are excluded from communication. The list is called authorized node-list (ANL) and contains information about the status of each node in the network, e.g. online, offline or revoked. The ANL is periodically updated and sent to each node.

The ANL includes nodes that are authorized (resembling a white-list) and nodes that are unauthorized (resembling a black-list) to participate in FIGO communication. The data structure is a list where entries have four attributes, as is shown in Table 4.

Table 4: Example of the node-list of N_i

Written node representation	Node ID	Status	Time-stamp	Key distribution value	List of common keys
N_j	902847710	Active	20100709 T 12:05	569.183	$K_{i,j}$, MK_j
N_g	891899456	Left	20100709 T 09:36	203.294	$K_{i,g}$, GK_g
N_k	897884821	Compromised	20100708 T 22:15	0	-

- **Written node representation:** How a node is presented in this thesis;
- **Node ID:** An identifier, e.g. the hardware-address of the node;
- **Status:** A status attribute indicating whether a node is active in the network, has left the network or has been revoked;
- **Time-stamp:** A time-stamp bound to the status indicating the time of the last received message about that node;
- **Key distribution value:** A value indicating which key the node currently has. More details over the key distribution value is given in Subsection 5.2.1;
- **List of common keys:** The list maintains which keys are shared with the nodes. $K_{i,j}$ is the pairwise key between N_i and N_j , MK_j is a mesh key generated by N_j and GK_g stands for the group key generated by N_g. Key material is explained in detail in Section 4.3.

Local controller *(LC)*: To prevent the *GC* from contacting every node in the network to distribute the group credential and ANL, local controllers are introduced. Mobile nodes try to connect with the *GC*. A selection of these nodes are chosen to become local controllers. Nodes that are not chosen will not utilize their connection with the *GC*.

An *LC* is chosen, based on the cost of the connection with the *GC*. The costs can be connection throughput or link stability. It is important to note that the *LC* is not appointed based on its position in the FIGO mesh, i.e. the number of neighbors is not taken into account.

An *LC* receives the group credential from the *GC*, compute the group key and is responsible to distribute the group key to its neighbors. Based on ANL, the *LC* notifies its group members of a revoked node.

Group members *(GM)*: A mobile node that has not used its connection with the *GC* is considered a group member. Group members have the responsibility to forward the group key to their direct neighbors.

From our network model, the question arises what happens when FIGO meshes merge. Since the most desired state of the mesh is when it is connected with the *GC*, it is clear that a mesh with no *LC* joins a mesh that has an *LC*. When two meshes with no *LC* merge, they form one big mesh without an *LC*. The details of this operation are specified in Subsection 5.4.4. The last possibility is when two meshes with an *LC* are able to merge. When this occurs the *LC* that has the "cheapest" connection with the *GC* will remain *LC*, and the other *LC* will become a *GM*. Since this is a costly operation in terms of key re-distribution, it is not advised to merge meshes with two *LC*s often.

4.2 Overview of KERS

The KERS protocol is divided in phases. Based on these phases, a node is in a state and a certain trigger causes the node to transfer to a different state. This section walks through the scheme phase by phase, and gives a high-level overview of the goal of each phase and describes how it is accomplished. The state chart in the following figure serves as a roadmap of this section.

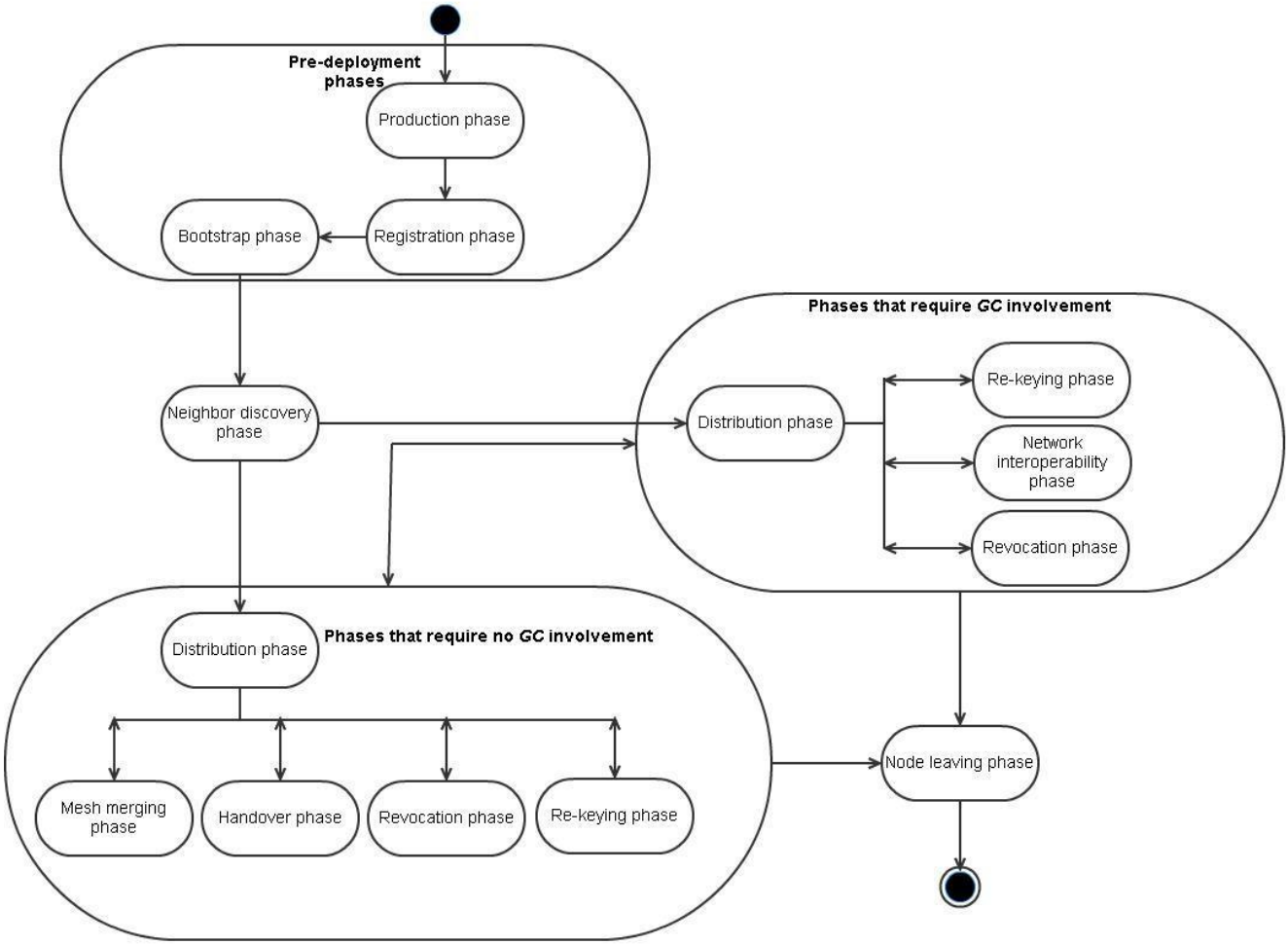

Figure 5: UML state chart of KERS

Figure 5 shows five different main states. The pre-deployment state consists of phases that take place before an incident occurs. When an incident occurs emergency response vehicles assemble at the incident site, and the mobile nodes that are mounted

in the vehicles start to perform neighbor discovery. After this is completed, mobile nodes try to establish a connection with the back office to engage in protocols of KERS that require involvement of the *GC*. Regardless of a connection with the *GC*, mobile nodes will autonomously engage in communications, establishing a secure ad-hoc network. These protocols are described in the phases that require no *GC* involvement. When an emergency vehicle leaves the incident site, the mobile node disconnects from the FIGO mesh. After the sensitive material is deleted, the node is considered to have exited the KERS session.

Mobile nodes can switch any time between states that requires *GC* involvement to a state that does not. For example, when a mobile node completes the distribution phase with the *GC*, it can enter the distribution phase with a mobile node. This is important, since it lifts restrictions about the sequence in which phases must be completed. A general overview of each phase is given to clarify the state chart and show the relations between different phases.

Production phase: When a node is manufactured, it receives cryptographic primitives. The node also receives a certificate, signed by the production company, and a group credential. The certificate proves the node can function in a FIGO network and the group credential is used for message authentication. The node can create a message authentication code to communicate with the *GC*.

Registration phase: The registration phase is triggered when a node is allocated to an emergency service vehicle. Registration occurs at the back office of the corresponding department. During the registration phase, a human administrator registers the node at the back office node. The node receives the credentials to authenticate itself to other nodes.

Bootstrap phase: The bootstrap phase serves to exchange network parameters related to the transmission of control traffic. The bootstrap phase is the last pre-deployment phase.

Neighbor discovery phase: When an incident occurs, emergency response vehicles will assemble at the incident area and perform neighbor discovery. The neighbor discovery phase serves two purposes: establishing a connection with the back office, and establishing a secure channel with other mobile nodes. The connection with the back office enables efficient revocation, because the back office has a global overview of the network. As a result of the neighbor discovery phase, nodes know that either the BON or a mobile node is responsible for key generation.

Distribution phase: The goal of the distribution phase is to provide all nodes with a group key for broadcast communication. When there is a *GC*, it generates and distributes preliminaries to mobile nodes to compute the group key. When there is no *GC*, the mobile node with the smallest key distribution value (see Subsection 5.2.1) is appointed as an *LC* and generates and distributes a mesh key. The responsibility of key distribution is given to multiple nodes. This increases the likelihood that all nodes receive the group key within a reasonable time.

Re-keying phase: The goal of this phase is to provide all nodes with fresh key materials, while avoiding network partitioning. An update schedule ensures that each node knows when a new group key is sent or must be created. The schedule prevents unauthorized key updates. Additionally, each group key has a configurable lifetime, depending on the confidentiality level of the data.

Revocation phase: The goal of the revocation phase is to revoke compromised nodes and keys. When a key is compromised, its confidentiality can no longer be guaranteed and the key must be revoked immediately. Second, when a node is captured, it must be placed on the revocation list.

Network interoperability phase: When the *GC* is connected to the FIGO mesh, different emergency service networks can merge when the back office authorizes the merge, which is called network interoperability. The network operability phase ensures that a new group key is distributed to all participating nodes.

Mesh merging phase: When nodes that are registered to the same *GC* have established different ad hoc mesh networks, they can merge. Initially both networks have different mesh keys and the mesh merging phase, ensures only one key will be used.

Handover phase: When an emergency vehicle moves around the incident site, the mobile node might connect to a different *LC* due to connectivity problems, this is called a handover. Since different *LC*s have different group keys, a node performing a handover must acquire the key of the *LC* it joins.

Node addition phase: When a FIGO mesh has been deployed and another emergency vehicle arrives at the incident area, the mobile node must join the existing network. The node addition phase is performed without intervention of the *GC*. The phase is an optimized instance of the initialization protocol between two mobile nodes.

Node leaving phase: When an emergency vehicle permanently leaves an incident area, its mobile node is notified. The notification is sent when the vehicle has a connection with the back office. The back office sends a signal to the mobile node, which causes the node to delete its key material. That a vehicle leaves the incident site can be decided by the emergency responder that drives the vehicle, but the node can also decide itself. In the latter, the node decides this based on the absence of any neighboring node for a certain period of time, say five minutes. Additionally, the *GC* can indicate that no further communication is needed and send a “kill” command to the node, resulting in the deletion of no longer needed sensitive material.

Now it is clear what the goal of each phase is and how it relates to other phases. This section showed that some phases do require interactions with the *GC*, while other phases do not, which makes the scheme robust for unstable connections.

4.3 Credentials and Message Compositions in KERS

In order to understand the functionality of our scheme, we present an overview of the

credentials and message compositions used in KERS. Credentials are values shared by entities that enable secure communications. In KERS, there are nine such credentials. Table 5 shows the credentials and their use.

Table 5: Overview of credentials in KERS

$cert_i$	Plain certificate of N_i
$cert_i^{GC}$	Signed certificate of N_i, by *GC*
certchain	Certificate chain containing information about the group a node belongs to
K_{MAC}	Key used in the algorithm for the message authentication code
$MAC_{i,j}$	Pairwise key used in the algorithm for the message authentication code
MAC_{GROUP}	Key used in the algorithm for the broadcast message authentication code
PKi	Public key of N_i
SKi	Private key, or signing key, of N_i
Ki,j	Symmetric pairwise key, shared between N_i and N_j
{M}K	Message *M* encrypted with key *K*
$(M)K_{MAC}$	Message authentication code over message *M*, with key K_{MAC}
DS_i	Device secret of local controller LC_i, input parameter to compute GK_i
NS	Network secret sent by group controller, input parameter to compute *GK*
$H(NS, DS_i)$	Hash function over secret *K* and the ID of LC_i to produce a unique *GK*
GK_i	Symmetric group key used for the network controlled by local controller LC_i
MK_i	Symmetric mesh key, generated by N_i

The first group of credentials consists of authentication credentials. Nodes prove their authenticity by exchanging verifiable certificates. N_i has $cert_i^{GC}$ to prove it is a legitimate node, belonging to an emergency response vehicle. The manufacturer and *GC* sign the certificate. Next, N_i possesses a key, K_{MAC}, used in the algorithm for the message authentication code (MAC). The key is used in communications between N_i and *GC* and preserves data integrity. Additionally N_i can establish a $MAC_{i,j}$ used for

MAC between N_i and N_i. Third, N_i can create or receive a MAC_{GROUP} that serves as key for the broadcast MAC. More detail on MACs can be found in Appendix A. Finally, N_i possesses a public/private key pair; both keys have a size of 256 bits to support a 128 pairwise key [48]. The private key is used to sign messages and is therefore referred to as signing key, SK_i. The public key, PK_i, can be used to verify the signature made by SK_i. N_i generates PK_i and SK_i, based on elliptic curve Diffie-Hellman (ECDH) and the key pair is refreshed each incident.

The second group of credentials consists of credentials that are used to generate encryption keys. The network secret (*NS*) is a credential that is used to compute the group key. *NS* is a random value generated by *GC* and distributed to *LC*s. The lifetime of *NS* is configurable at the back office. The device secret (*DS*) is the second credential used to compute the group key. It is a random value that is unique for each node. The lifetime of *DS* is set during the configuration phase. Additionally, the public/private key pair is also used as parameter for the pairwise key.

The third group of credentials consists of encryption keys to provide data confidentiality. The pairwise key ($K_{i,j}$) is the result of the neighbor-discovery phase. Using ECDH N_i and N_j compute $K_{i,j}$ by computing

$$hPK_iSK_j = hPK_jSK_i,$$

Where h is a domain parameter for ECDH, PK_i is N_i its public key and SK_j is N_j its private key. More information on ECDH can be found in Appendix A. $K_{i,j}$ is used for secure unicast communications, like transferring the group key from N_i to N_j. Sine the key pair of a node is refreshed every incident, the pairwise key has the same refresh rate. The group key (*GK*) is a key that enables nodes in a FIGO network to securely use broadcast communications. *GK* is a joined effort between group controller (*GC*) and local controller (*LC*). It is computed based on the device secret of *LC* and the *NS* that is generated by *GC*. *LC* uses a hash over these two values, which results in *GK*. By setting the lifetime of *NS* equal to the lifetime of *GK* the lifetime of *GK* is

configurable by the back office, dependent on the desired confidentiality level of the data that is transmitted during the incident. The mesh key (*MK*) is generated when *GC* is unable to distribute *NS* to the network. It is a random value and the lifetime is fixed during the configuration phase or a software update.

We advise that key lengths for AES generated keys have a size of 128-bit [34]. To create an equally secure pairwise key, the ECDH public-private key pair must consist of keys with a size of 256 bit [48]. With regard to the MAC we advice to use HMAC [10][11]. HMAC is based on SHA-1 [32], which produces a 160-bits message digest and is equal to the output of the HMAC algorithm. The key size of the HMAC algorithm is advised to be between 160/2 and 160 [35]. The key is a symmetric key and therefore we advise a key of size 128-bit.

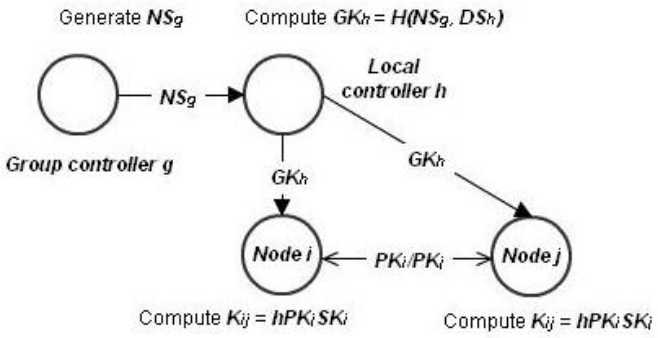

Figure 6: Key material when there is a connection with *GC*

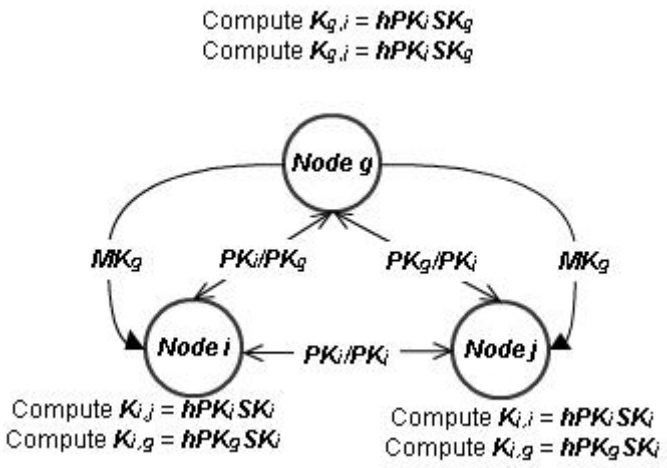

Figure 7: Key material when there is *no* connection with *GC*

Messages in KERS are used to transfer key material, which is confidential. All messages have a label, e.g. REGISTER, HELLO or STATUS. KM_{REGISTER} refers to the key management message with label "REGISTER". In KERS, there are nine different message types. The content of each message is given in the protocol descriptions. Here we give a brief overview of the purpose of each message.

- ***KM*$_{\text{REGISTER}}$:** Is used during the registration phase and N_i sends this message to a laptop or computer to indicate it wants to be registered with *GC*.
- ***KM*$_{\text{SIGN}}$:** Forwards the registration request of N_i to *GC*.
- ***KM*$_{\text{SIGNED}}$:** The reply from *GC* to a KM_{SIGN} message, indicating that the register request has been processed successfully. The message contains a signed certificate for N_i.
- ***KM*$_{\text{REGISTERED}}$:** The message forwards the KM_{SIGNED} message to N_i. The message includes additional credentials that were not present in the KM_{SIGNED} message.
- ***KM*$_{\text{HELLO}}$:** A message indicating that the sender is looking to establish secure channels with its neighbors.
- ***KM*$_{\text{SETUP}}$:** The message indicates that the sender has a group key for confidentiality and data integrity protection. The message is sent over the secure unicast channel.
- ***KM*$_{\text{INTEROP}}$:** Is sent to indicate that the FIGO network is going to merge with

another FIGO network and that the key in the message should be used for broadcast communications that are sent from one network to the other.

- ***KM*$_{\mathbf{STATUS}}$:** A message sent during the neighbor discovery phase. It indicates the key distribution value. More information about the key distribution value can be found in Section 5.4.
- ***KM*$_{\mathbf{ADHOC}}$:** Is the equivalent of KM_{SETUP} for the mesh key and broadcast MAC key. Additionally it includes the key distribution value of the sender.

This concludes the section that introduced KERS to the reader. The network model is adapted to support distributed key distribution and clustering of the network. We gave a schematic overview of KERS, which showed that the scheme has different phases. Nodes must follow a specific route in KERS, however parallelism between interaction with a mobile node or *GC* enables flexible key management. Finally, this section presented which credentials are used in KERS and what messages are sent during key management operations. Now that we have presented the general aspects of KERS, in the next chapter we present the detailed description of KERS.

5. Detailed Protocol Description of KERS

In this chapter, we present our key management scheme for emergency response systems (KERS). The focus of our solution is on improving the key management scheme of FIGO. The scheme focuses on the weaknesses of FIGO, described in Section 2.4.

The original key management scheme of FIGO does not support dynamic key updates and does not use separate keys for separate security mechanisms. The main disadvantage is that over-the-air re-keying is not supported.

Another disadvantage is that the back office node (BON) is the only entity to perform key distribution, which introduces a single point of failure. In Chapter 4, we introduced KERS and proposed a network model allowing mobile nodes to distribute group credentials, mitigating the single point of failure.

KERS provides dynamic key updates and unicast communications to distribute the group key. A network hierarchy supports member dynamics and appoints nodes with the task of group key generation and distribution. KERS specifies a key update mechanism that minimizes the chance of network partitioning. It also describes an algorithm that ensures that nodes will minimize the amount of control traffic sent by nodes.

Section 5.1 presents protocols of KERS prior to node deployment at an incident site. Section 5.2 describes the neighbor discovery protocol and provides details on control traffic in KERS. Section 5.3 presents protocols of KERS requiring involvement of a group controller (*GC*). Finally, Section 5.4 describes protocols of KERS that do not require involvement of *GC*.

5.1 Key Management Phases Prior to Deployment

There three phases that occur prior to deployment are the *production*, *registration* and *bootstrap* phase. When a node completes the production phase, it possesses credentials needed to be registered during the registration phase. The bootstrap phase

provides a node with parameters for exchanging control traffic.

Each phase is briefly introduced by describing the goal of the phase and the message sequence chart (MSC) of the protocol. When MSCs do not provide additional clarity, it is omitted. Next, a detailed message description of the protocol is given describing the message contents and in detail.

5.1.1 Production Phase

The production phase is where the node is produced. During the production phase, nodes receive primitives for cryptographic operations and authentication. It is assumed there are no security threats during the production phase. As a result, the production phase consists of a single communication from a master device, *MD*, to the mobile node, N_i. The master device is a device that issues certificates from the production company. The protocol signals to the node that it has finished the production phase and can be issued to emergency services. Due to the simplicity of this phase, an MSC is omitted.

Production phase protocol:

$MD \rightarrow N_i$: $ID_{MD}||ID_i||\text{PRODUCED}||cert_i^{PROD}||PK_i||K_{MAC}$

N_i can prove its identity to the *GC*, which is necessary for the next phase. The node can perform elliptic curve Diffie-Hellman and use a message authentication code, with a key that is shared with the *GC*.

5.1.2 Registration Phase

During registration, the *GC* registers nodes to its network domain. When the registration is successful, nodes possess credentials to authenticate themselves.

The registration phase is the only phase requiring human intervention, by an administrator, e.g. the IT manager of the emergency service department. An administrator uses the web interface (*WEB*) to register nodes to *GC*. It is assumed that

WEB and *GC* share a secure channel, because both entities share other secure communications besides KERS.

As a result of the registration phase N_i possesses a certificate chain (*certchain*) to authenticate itself to nodes belonging to a different emergency response network. Additionally, N_i possesses a certificate ($cert_i$) to authenticate itself to nodes belonging to the same network. The public key of *GC* is used to verify the authenticity of other nodes in KERS. The registration protocol is illustrated by the message sequence chart (MSC) in the following figure, where details of decryption and verification are omitted for readability. The details of the protocol are presented in Figure 8.

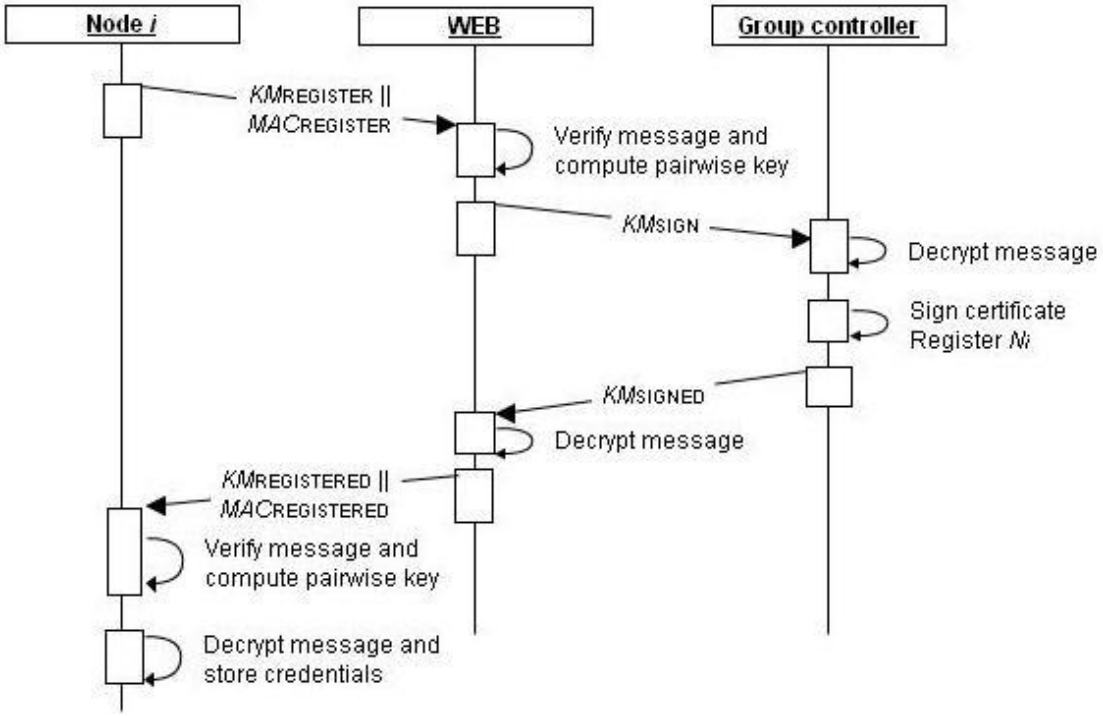

Figure 8: MSC of the registration protocol

Registration protocol:

$N_i \rightarrow WEB$: $KM_{\text{REGISTER}} = ID_i||ID_{WEB}||\text{REGISTER}||cert_i^{\text{PROD}}||PK_i||$
$MAC_{\text{REGISTER}} = (KM_{\text{REGISTER}})K_{\text{MAC}}$

WEB : Use K_{MAC} to verify MAC_{REGISTER}
if verification holds PROCEED,
else ABORT;

Verify $cert_i^{\text{PROD}}$ with PK_{PROD},
if verification holds PROCEED,
else ABORT;

Compute $K_{WEB,i} = hPK_iSK_{WEB}$ (see Appendix A)
where h is, PK_i is the public key from N_i
and SK_{WEB} the private key from WEB.

$WEB \rightarrow GC$: $ID_{WEB}||ID_{GC}$ ||SIGN||$\{cert_i\}K_{WEB,GC}$

GC : Decrypt $\{cert_i\}K_{WEB,GC}$ with pairwise key $K_{GC,WEB}$

Sign $cert_i$, resulting in $cert_i^{GC}$,

Register N_i is with GC.

$GC \rightarrow WEB$: $KM_{\text{SIGNED}} = ID_{GC}||ID_{WEB}||\text{SIGNED}||\{PK_{GC}||(cert_i)SK_{GC}||$
$certchain\}K_{GC,WEB}$

WEB : Decrypt KM_{SIGNED} with pairwise key $K_{WEB,GC}$.

$WEB \rightarrow N_i$: $KM_{\text{REGISTERED}} =$
$ID_{WEB}||ID_i||\text{REGISTERED}||PK_{GC}||\{PK_{GC}||cert_i^{GC}||$
$certchain||cert_{WEB}^{PROD}\}K_{WEB,i}$

$MAC_{\text{REGISTERED}} = (KM_{\text{REGISTERED}})K_{\text{MAC}}$

N_i : Use K_{MAC} to verify $MAC_{\text{REGISTERED}}$
if verification holds PROCEED,
else ABORT;

Compute $K_{i,WEB} = hPK_{WEB}SK_i$ (see Appendix A)
where h is, PK_{WEB} is the public key from WEB
and SK_i the private key from N_i;

Decrypt $KM_{REGISTERED}$ with pairwise key $K_{i,WEB}$;

Verify $cert_{WEB}^{PROD}$ with PK_{PROD},
if verification holds PROCEED,
else ABORT;

Store PK_{GC}, $cert_i^{GC}$ and *certchain*.

The result of the registration phase is that N_i has credentials to build up secure channels with *GC* and nodes from the same FIGO network. N_i can use its message authentication code in communications with nodes from the same FIGO network.

5.1.3 Bootstrap Phase

Besides credentials, the *GC* and a mobile node exchange configuration parameters. These can include network addresses or parameters. In theory, we separate the bootstrap phase from the registration phase for clarity. Nodes can be registered to a *GC*, but without passing the bootstrap phase, they cannot be deployed.

The parameters that are exchanged for KERS are related to the neighbor discovery phase, see Section 5.2.1. Here we only give the protocol of the bootstrap phase. Because the protocol is simple, the MSC is omitted here.

Bootstrap protocol:

$N_i \rightarrow WEB$: $KM_{BOOTSTRAP} = ID_i || ID_{WEB} ||$ BOOTSTRAP$|| cert_i^{GC}$
$MAC_{BOOTSTRAP} = (KM_{BOOTSTRAP}) K_{MAC}$

WEB : Use K_{MAC} to verify $MAC_{BOOTSTRAP}$
if verification holds PROCEED,
else ABORT;

Verify $cert_i^{GC}$ with PK_{GC},
if verification holds PROCEED,
else ABORT;

$WEB \rightarrow GC$: $KM_{BOOTSTRAP}$

$GC \rightarrow WEB$: $ID_{GC} || ID_{GC} ||$ PARAM$|| \{r_{mesh}, r_{GC}\} K_{GC,WEB}$

$WEB \rightarrow N_i$: KM_{PARAM}

$MAC_{PARAM} = (KM_{PARAM}) K_{MAC}$

N_i : Use K_{MAC} to verify MAC_{PARAM}
if verification holds PROCEED,
else ABORT;

Decrypt KM_{PARAM} with pairwise key $K_{i,WEB}$;

Store parameters.

5.2 Key Management Phase for Neighbor Discovery

When a node is deployed, it must first establish if other FIGO nodes are nearby. When a node has established connections with its neighbors, it is part of a network.
In KERS, nodes exchange information about the status of the network. A network can be connected to a *GC*, but it does not need to be. However, a connection with *GC* is desirable, because it enables the back office to exert some control over the network.

To indicate whether a node has a key and if this is a group key or mesh key, KERS introduces a *kd*-value. This value enables the node to exchange status information about the state of the network.

5.2.1 Exchanging Status Information

Status information is communicated through the *kd*-value. The *kd*-value falls into different intervals, bounded by random values r_{GC} and r_{mesh}. The values r_{GC} and r_{mesh} are generated, updated and transferred by *GC*.

When N_i is deployed it randomly generates $kd_i < r_{mesh}$, which is bound to the state of N_i. There are three states when a node is deployed:

1. **No key state:** If $kd_j < r_{mesh}$, then N_j has no group or mesh key. N_i will generate a mesh key if $kd_i < kd_j$ for all incoming kd_j and change kd_i to $r_{mesh} < kd_i < r_{GC}$. If $kd_i > kd_j$ for at least one incoming kd_j, N_i will remain in the no key state;
2. **Mesh key sate:** If $r_{mesh} < kd_j < r_{GC}$, then N_j has a mesh key. N_i will connect to N_j, obtain the mesh key and set $kd_i=kd_j$;
3. **Group key state:** If $kd_j > r_{GC}$, then N_j has a group key (*GK*) and there is a connection with *GC*. N_i connects to N_j, receives *GK* and sets $kd_i=kd_j$.

N_i indicates its status to neighboring nodes by sending a KM_{STATUS} message containing kd_i. The response of neighboring nodes is dependent of the state of the network and value of kd_i. To limit the times of exchanging of status information, the following rules apply in KERS:

- A node in the group key state (*state 3*) does not send KM_{STATUS} messages
- A node in the mesh key state (*state 2*) does not send KM_{STATUS} messages to nodes in *state 1*
- A node in the no key state (*state 1*) sends KM_{STATUS} messages to all its neighbors

Nodes in *state 2* send KM_{STATUS} messages less frequent than nodes in *state 1*. A node in *state 3* only receives KM_{STATUS} messages.

When the state of a node changes to a higher state, e.g. from *state 1* to *state 2*, it is because they receive or generate a key. The node forwards the key to its neighbors, causing them to change sates and preventing message overhead, by limiting control traffic.

5.2. 2 Neighbor Discovery Phase

In the neighbor discovery phase, nodes establish secure channels with their direct neighbors without requiring an existing communication infrastructure. A connection between N_i and the *GC* does require such an infrastructure and due to this difference, the neighbor discovery phase has two protocols.

In both cases, N_i starts broadcasting KM_{HELLO} messages containing the verifiable part of the certificate and public key of N_i. The broadcast message is single-hop and enables the receiver to authenticate N_i and to compute the pairwise key.

When the *GC* receives KM_{HELLO}, it replies with its certificate to N_i. When N_i successfully verifies the certificate, it initiates a VPN connection with the *GC*. Next, N_i waits for the group secret to be sent.

Neighbors of N_i also broadcast their KM_{HELLO} message. Upon receiving such a message from N_j, N_i verifies the certificate and computes the pairwise key. N_i sends a KM_{STATUS} message to N_j encrypted with the pairwise key indicating that N_i has received the KM_{HELLO} message. Additionally, KM_{STATUS} includes the *kd*-value of N_i. N_j decrypts the KM_{STATUS} message and based on the *kd*-value it sends its key or waits.

The protocols for neighbor discovery with the *GC* and mobile nodes are described separately. The neighbor-discovery protocol for two mobile nodes is illustrated in an MSC. The MSC of the neighbor discovery phase between the *GC* and N_i is omitted, due to the simplicity of the protocol.

When the neighbor discover phase is completed, N_i has established secure channels with authenticated neighbors. The secure channel can be used to transport a group or mesh key.

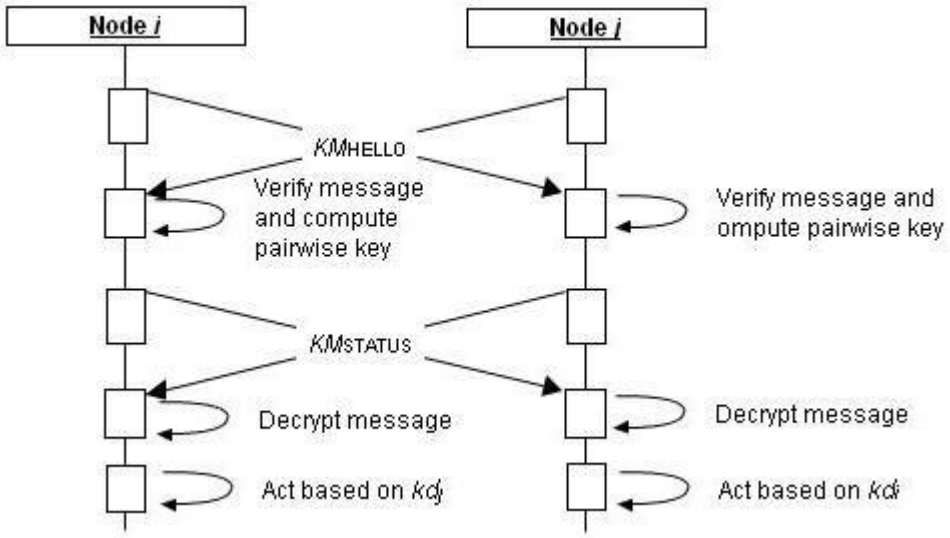

Figure 9: MSC of the neighbor discovery protocol between nodes

Neighbor discovery protocol between N_i and N_j:

$N_{i,j} \rightarrow *$: $KM_{\text{HELLO}} = ID_i || ID_* ||$HELLO$|| cert_i^{GC} || PK_i || (ID_i || PK_i) SK_i$

$N_{i,j}$: Verify KM_{HELLO} with PK_j,
if verification holds PROCEED,
else ABORT;

Verify $cert_j^{GC}$ with PK_{GC}
if verification holds PROCEED,
else ABORT;

Compute $K_{i,j} = hPK_jSK_i$ (see Appendix A)
where h is, PK_j is the public key from N_j
and SK_i the private key from N_i.

$N_i \rightarrow N_j$: $KM_{\text{STATUS}} = ID_i || ID_j ||$STATUS$|| NONCE_i ||$
$\{kd_i || NONCE_i || MAC_{i,j}\} K_{i,j}$

$N_j \rightarrow N_i$: KM_{STATUS}

$N_{i,j}$: Decrypt KM_{STATUS} with pairwise key $K_{j,i}$,
if $NONCE_i$ is fresh and unaltered, then PROCEED,
else ABORT;

Based on kd_i wait for GK or MK, or
generate MK.

Neighbor discovery protocol between N_i and GC:

$N_i \rightarrow *$: $KM_{\text{HELLO}} = ID_i||ID_*||\text{HELLO}||cert_i^{GC}||PK_i||(ID_i||PK_i)SK_i$

GC : Verify KM_{HELLO} with PK_i,
if verification holds PROCEED,
else ABORT;

Verify $cert_j^{GC}$ with PK_{GC}
if verification holds PROCEED,
else ABORT;

$GC \rightarrow N_i$: $KM_{\text{GCONLINE}} = ID_{GC}||ID_i||\text{GCONLINE}||\text{NONCE}||$
$(\text{NONCE}||\text{cert}_{\text{GC}}^{\text{GC}})PK_i$

N_i : Decrypt KM_{GCONLINE} pairwise key $K_{i,GC}$,
if $NONCE_i$ is fresh and unaltered, then PROCEED,
else ABORT;

Verify $cert_{GC}^{GC}$ with PK_{GC}
if verification holds PROCEED,
else ABORT;

Establish OpenVPN connection with GC.

Wait for the network secret from the GC.

5.3 Key Management Phases with GC Involvement

To distribute the group key (*GK*), first *GC* generates a network secret (*NS*) that serves as seed for *GK*. Next, *GC* distributes *NS* to the local controllers (*LC*s), which have their own device secret (*DS*), serving as the second parameter to compute *GK*. Each sub-group in the emergency response network has a different *GK*.

Each *GK* has an expiry time that limits the damage an attacker can cause by compromising the key. The *GC* updates and distributes the *NS* at a configurable

interval. Due to unstable connections, the *NS* cannot always be sent to *LC*s. Alternatively, the *LC* can update its *DS* to update *GK*.

Communication between different emergency services is only supported when both networks are connected to their back office. The back offices agree on a single network interoperability key for secure communication. After distribution of this key, both networks can communicate.

5.3.1 Group Key Distribution Phase

The *GC* will exclusively distribute the *NS* to LC_j that computes the GK_j by hashing the *NS* and the device secret DS_j. LC_j distributes GK_j to its group members using the pairwise key. The MSC is shown in Figure 10.

To detect replay attacks NONCEs are included in the messages. A NONCE is fresh when it is not used before. More information on NONCEs can be found in Appendix A.

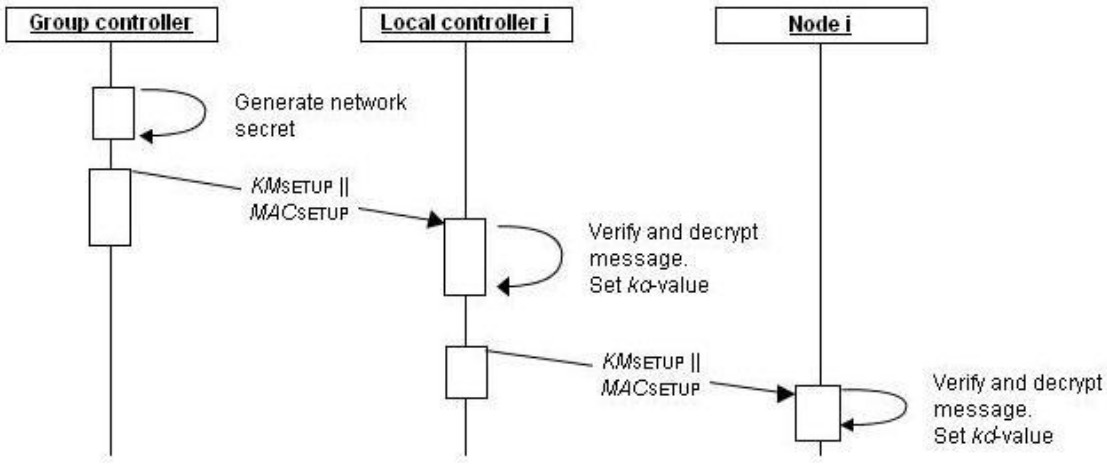

Figure 10: MSC of key distribution when *GC* is online

Distribution protocol when GC is online:

GC: Generate network secret NS_{GC}

$GC \rightarrow LC_j$: $KM_{SETUP} = ID_{GC} || ID_j || SETUP || \{MAC_{GROUP} || NS_{GC}\} K_{GC,j}$

$MAC_{SETUP} = (KM_{SETUP}) K_{MAC}$

LC_j : Use K_{MAC} to verify MAC_{SETUP}
if verification holds PROCEED,
else ABORT;

Decrypt KM_{SETUP} with pairwise key K_{jGC};

Compute $GK_j = H(NS_{GC}, DS_j)$;

Set $kd_j > r_{GC}$.

$LC_j \rightarrow N_i$: $KM_{SETUP} || MAC_{SETUP}$

N_i : Use $MAC_{i,j}$ to verify MAC_{SETUP}
if verification holds PROCEED,
else ABORT;

Decrypt KM_{SETUP} with pairwise key $K_{i,j}$;

Set $kd_i = kd_j$.

Since all communications take place over the secure VPN channel, we did not provide a NONCE in the message exchange.

5.3.2 Re-keying phase

Key updates are issued before the current key expires to minimize the chance of network partitioning. Time-slots are allocated to update the network secret (*NS*) or device secret (*DS*) as shown in Figure 11.

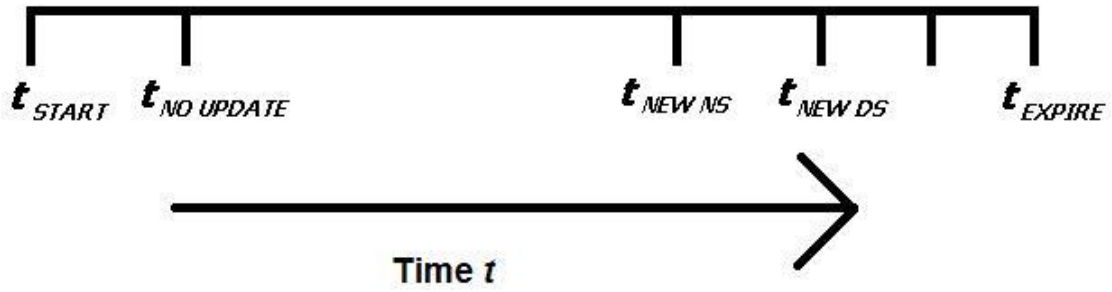

Figure 11: Timeline of *GK*

When there is a connection between *GC* and local controller N_i (LC_i), group key GK_i is updated at $t_{new\ NS}$ and distributed with pairwise key $K_{GC,i}$. When the connection between *GC* and LC_i is unavailable from $t_{new\ NS}$ until $t_{new\ DS}$, LC_i refreshes DS_i, and computes a new GK_i with the non-updated *NS*. When the connection with *GC* is restored, LC_i notifies *GC* that it updated DS_i, and communicates the expiry date of the updated GK_i.

When compromised node N_c is moved to the revocation list and the lifetime of *GK* has not reached $t_{new\ NS}$, LC_i receives a new *NS* and distributes the updated GK_i as prescribed by the distribution phase.

Due to a mismatch in clock synchronization it may be possible that two keys are active at some point in time. The key which is used depends on which input parameter is used. If the key is generated based on an updated *NS*, it precedes all other keys. Meaning that if a key is generated by LC_i, based on an updated DS_i and an expired *NS*, while *GC* updated its *NS*, *LC* has to re-compute GK_i based on the updated *NS*.

The re-keying phase is described for updating the confidentiality key. It is assumed that together with an update of *GK*, MAC_{GROUP} is also updated.

5.3.3 Interoperability between Networks

An emergency response network from an emergency service might temporarily want to connect to a different emergency service. This is called network interoperability.

Security in the merged networks should be stronger or equal to security in the separate networks. Network interoperability is always coordinated by the back office.

When the networks merge, both meshes keep their own group keys (GK_p and GK_f) to secure communications within their respective network. Traffic between the networks is secured by $GK_{pf}=GK_{fp}$. The *GC*s from the networks agree on the group key and distribute it to their *LC*s. The network interoperability protocol is illustrated with a state diagram as shown in the Figure 12.

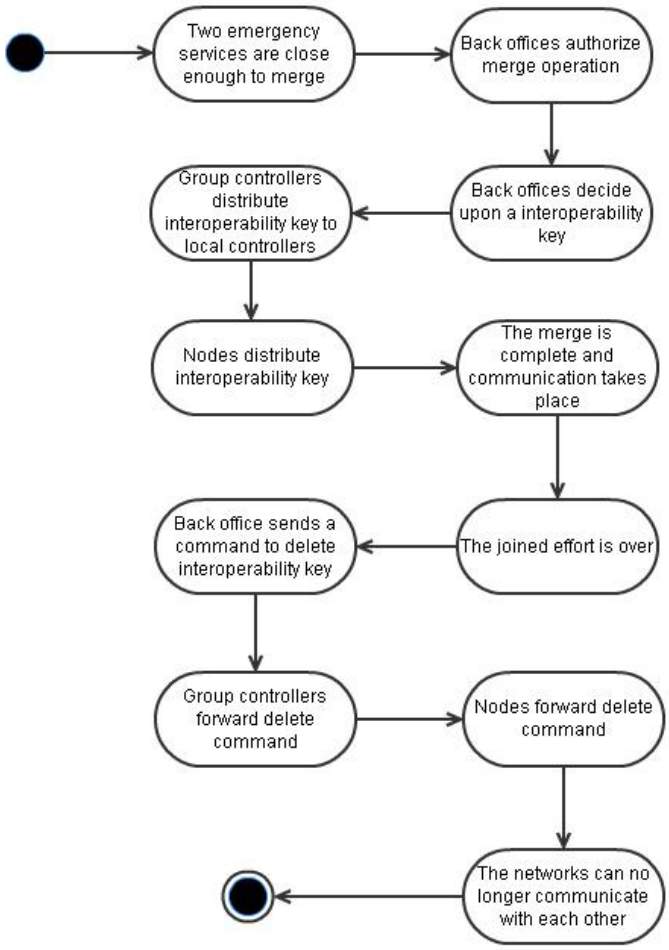

Figure 12: UML state chart for the network interoperability phase

Figure 13 gives a graphical representation of the connections that are formed during network interoperability. Nodes are subscripted with the first letter of the department they belong to, so N_{p1} is a mobile node with identifier 1, belonging to the

police department. Similar N_{f3} is a mobile node with identifier 3, belonging to the fire department.

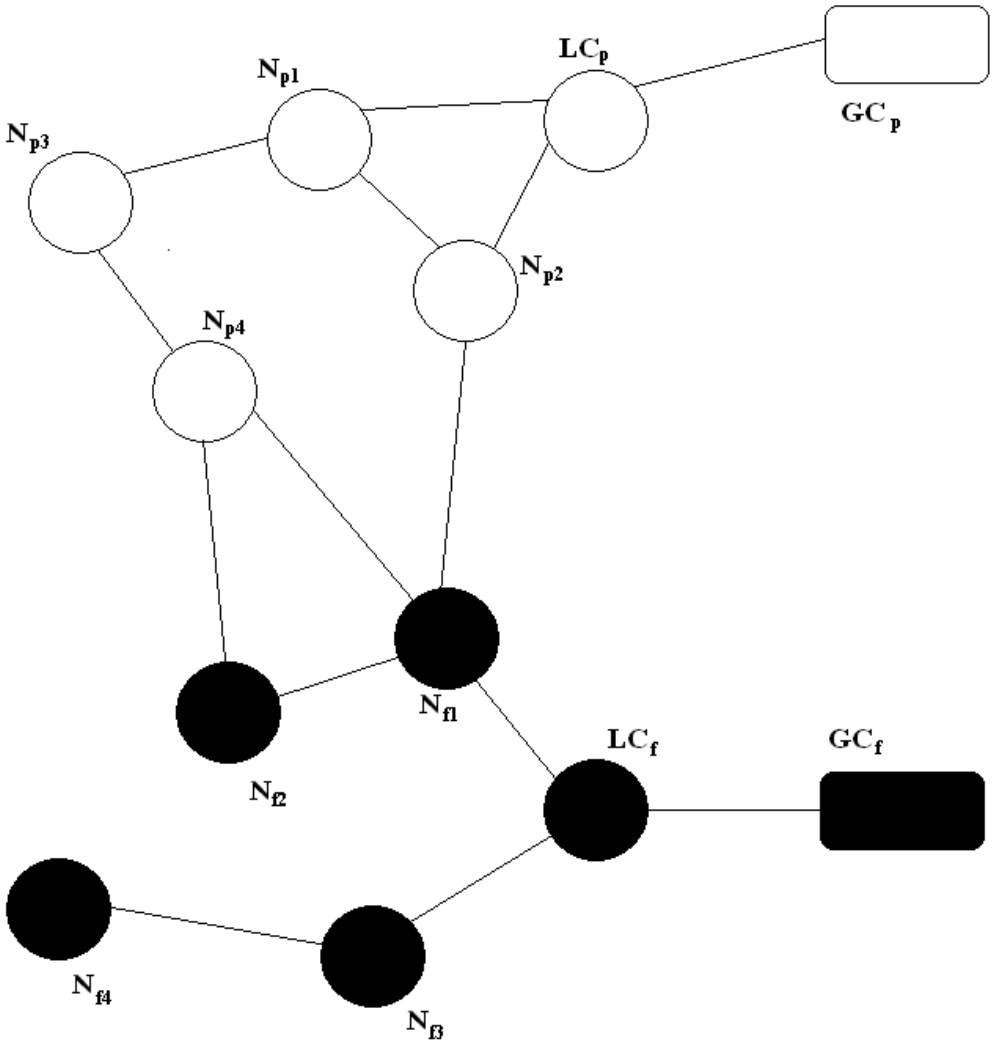

Figure 13: Graphical representation of the network interoperability phase

The message sequence chart of the network interoperability protocol is omitted, because it is identical to the key distribution phase that represented in Figure 6.

Network interoperability protocol:

$GC_p \rightarrow LC_p$: $KM_{\text{INTEROP}} = ID_{GCp} || ID_{LCp} ||$ INTEROP $|| \{MAC_{GROUP} || GK_{pf}\} K_{GCp,LCp}$
$MAC_{\text{INTEROP}} = (KM_{\text{INTEROP}}) K_{MAC}$

LC_p : Use K_{MAC} to verify MAC_{INTEROP}
if verification holds PROCEED,
else ABORT;

Decrypt KM_{INTEROP} with pairwise key $K_{LCp,GCp}$;

Store GK_{pf}.

$LC_p \rightarrow N_{p*}$: $KM_{\text{INTEROP}} || MAC_{\text{INTEROP}}$

N_{p*} : Use $MAC_{p*,LCp}$ to verify MAC_{INTEROP}
if verification holds PROCEED,
else ABORT;

Decrypt KM_{INTEROP} with pairwise key $K_{Np*,LCp}$;

Store GK_{pf};

Forward $KM_{\text{INTEROP}} || MAC_{\text{INTEROP}}$ to neighbors.

Communications between *GC* and *LC* take place over a secure VPN channel, so a NONCE is excluded from the message. However, when forwarding the KM_{INTEROP} and MAC_{INTEROP} messages, *LC*s and mobile nodes should include a NONCE to prevent replay attacks.

The split up of meshes should be coordinated from the back office, sending a message to *LC*s that indicates that the coordinated effort has stopped. *LC* deletes the interoperability key and sends a message to its neighbors, ordering them to delete the interoperability key and forward the message to their neighbors. Both services can continue to use the former group key for secure communications within their network.

5.4 Decentralized Key Management in KERS

Due to characteristics of the wireless medium used for data exchange, connections are unstable. The connection to the back office cannot be guaranteed. When the connection with the back office node is temporarily unavailable, key management should still function properly. This subsection shows the operation of KERS when there is no group controller (*GC*).

5.4.1 Key Distribution Phase

When nodes cannot connect with the *GC*, a mesh key (*MK*) is used for secure communication in the mesh. The *MK* is transported over the pairwise secured link between nodes.

The node with the smallest key distribution value (*kd*-value) generates the *MK*. A node is rarely able to reach all nodes in the mesh with single-hop communication. To avoid network partitioning the mesh key bound to the lowest *kd*-value propagates through the whole mesh.

After N_i receives the *MK* from N_j, N_i takes on the same *kd*-value as N_j. N_i forwards the *MK* to all direct neighbors with a higher *kd*-value. Eventually all nodes in the mesh possess the same *MK*.

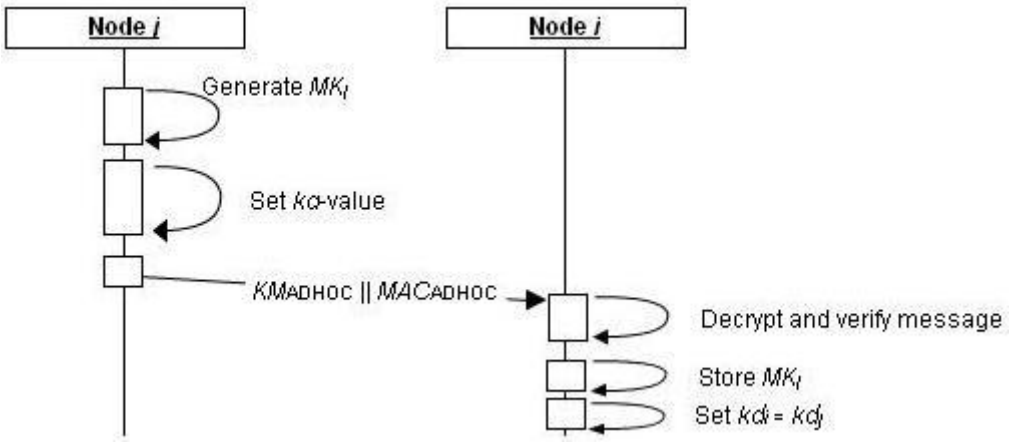

Figure 14: MSC for the key distribution phase without GC involvement

Distribution protocol with no GC involvement:

N_j : Generate and store MK_j;

Set $r_{MESH} < kd_j < r_{GC}$.

$N_j \rightarrow N_i$: $KM_{ADHOC} = ID_j || ID_i || ADHOC || \{MAC_{GROUP} || NONCE_j || kd_j || MK_j \} K_{j,i}$

$MAC_{ADHOC} = (KM_{ADHOC}) MAC_{i,j}$

N_i : Use $MAC_{i,j}$ to verify MAC_{ADHOC}
if verification holds PROCEED,
else ABORT;

Decrypt KM_{ADHOC} with pairwise key $K_{i,j}$,
if $NONCE_j$ is fresh and unaltered,
then PROCEED, else ABORT;

Store MK_j.

Set $kd_i = kd_j$.

5.4.2 Decentralized Re-Keying

Updating the *MK* is tedious, since there is no clock synchronization or entity with a global view of the network. The timeline in Figure 15 shows that the *MK* can be updated at different moments during its lifetime.

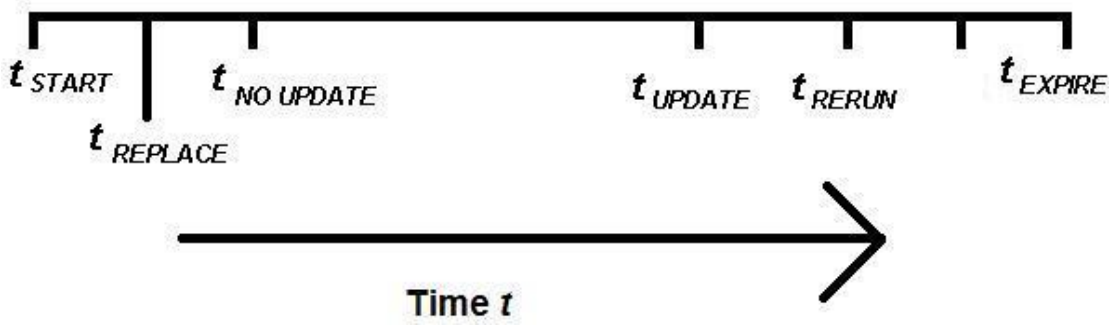

Figure 15: Timeline of *MK*

MK_i can be updated directly after it is generated when two emergency response networks merge after both networks acquired an *MK*. This situation is explained in more detail at the mesh merging protocol in sub-sub-section 5.6.4.

When N_i is connected to the emergency response network, MK_i gets updated at t_{update}. When N_i is no longer part of the network, the protocol for decentralized key distribution is initiated at t_{rerun}.

If MK_i expires and a new one is not received, nodes reset their *kd*-value such that it indicates they have no key. We assume that with an update of MK_i, MAC_{GROUP} is also updated.

5.4.3 Node Addition

When N_i is deployed it broadcasts KM_{HELLO} messages and neighboring nodes that already belong to a network, reply with a KM_{STATUS} message. This KM_{STATUS} message contains the certificate and public key of the sender, and the *kd*-value. After successful verification of the certificate, N_i can compute the pairwise key and decrypt the status information.

Based on the *kd*-value, N_i can decide to wait for other KM_{STATUS} messages or reply with a KM_{STATUS} message, requesting the group key (*GK*). The latter is the case when the *kd*-value indicates there is a connection with the *GC*. When N_j receives the KM_{STATUS} message from N_i it replies with a message containing the *GK*. The MSC is presented in Figure 16.

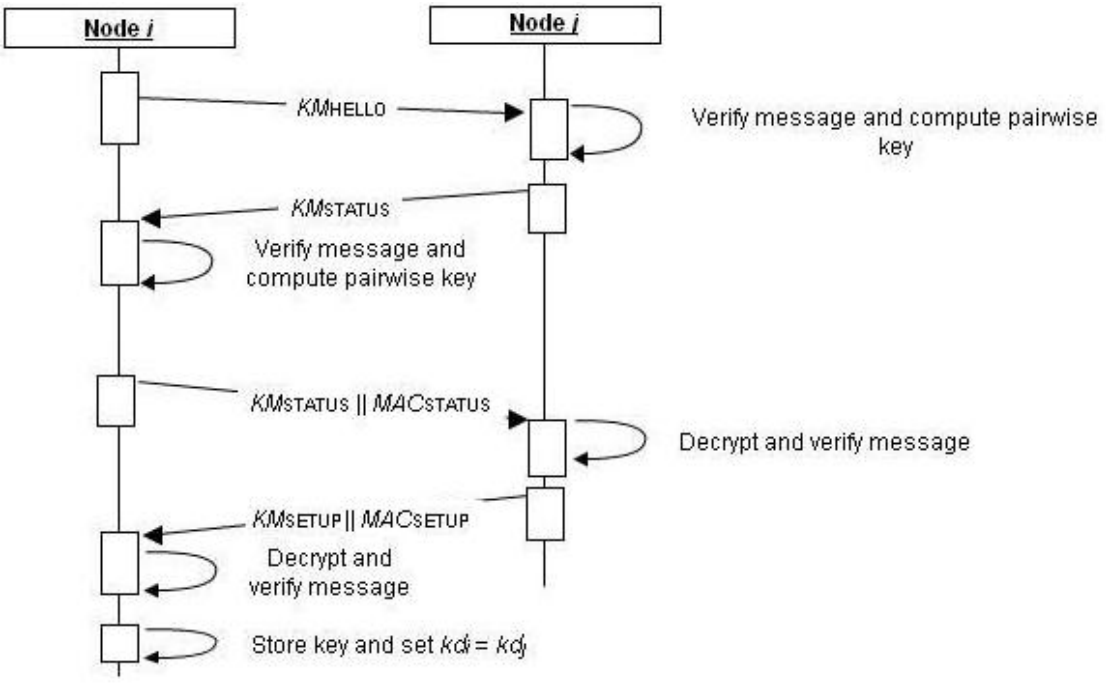

Figure 16: MSC of the node addition phase

Node addition phase:

$N_i \rightarrow *$: $KM_{\text{HELLO}} = ID_i||ID_*||\text{HELLO}||cert_i^{GC}||PK_i||(ID_i||PK_i)SK_i$

N_j : Verify KM_{HELLO} with PK_i,
if verification holds PROCEED,
else ABORT;

Verify $cert_i^{GC}$ with PK_{GC}
if verification holds PROCEED,
else ABORT;

Compute $K_{j,i} = hPK_iSK_j$ (see Appendix A)
where h is, PK_i is the public key from N_i
and SK_j the private key from N_j.

$N_j \rightarrow N_i$: $KM_{\text{STATUS}} = ID_j||ID_i||\text{STATUS}||PK_j||NONCE_j||$
$\{NONCE_j||kd_j||MAC_{i,j}\}K_{j,i}$

N_i : Compute $K_{i,j} = hPK_jSK_i$ (see Appendix A)
where h is, PK_j is the public key from N_j
and SK_i the private key from N_i.

Decrypt KM_{STATUS} with pairwise key $K_{i,j}$,
if $NONCE_j$ is fresh and unaltered,
then PROCEED, else ABORT;

$N_i \rightarrow N_j$: $KM_{\text{STATUS}} = ID_i||ID_j||\text{STATUS}||NONCE_i||\{NONCE_i||kd_i\}K_{i,j}$
$MAC_{\text{STATUS}} = (KM_{\text{STATUS}})MAC_{i,j}$

N_j : Use $MAC_{i,j}$ to verify MAC_{STATUS}
if verification holds PROCEED,
else ABORT;

Decrypt KM_{STATUS} with pairwise key $K_{j,i}$,
if $NONCE_i$ is fresh and unaltered,
then PROCEED, else ABORT;

Decide, based on s-value, to send the group/mesh key.

$N_j \rightarrow N_i$: $KM_{\text{SETUP}} = ID_j||ID_i||\text{SETUP}||NONCE_j'||\{NONCE_j'||$
$MAC_{GROUP}||KEY\}K_{j,i}$

```
      MAC_SETUP = (KM_SETUP) MAC_i,j

N_i   : Use MAC_i,j to verify MAC_SETUP
      if verification holds PROCEED,
      else ABORT;

      Decrypt KM_SETUP with pairwise key K_i,j ,
      if NONCE_j, is fresh and unaltered,
      then PROCEED, else ABORT

      Store KEY,

      Set kd_i=kd_j.
```

5.4.4 Mesh Merging Phase

When N_i and N_j belong to different networks, but are registered at the same BON the networks can merge. As opposed to the network interoperability phase, the merge of networks can occur without intervention of the *GC*.

Both networks have different mesh keys, i.e. MK_i and MK_j. The key bound to the lowest *kd*-value becomes the key for the merged network. Before nodes exchange their *kd*-values, they need to establish a secure channel.

When N_i receives a KM_{HELLO} message from N_j, it computes the pairwise key replies with a KM_{STATUS} message. After successful decryption, N_j also replies with a KM_{STATUS} message. Both nodes conclude on the incoming *kd*-value that the other node possesses a *MK*. When $kd_i<kd_j$, N_i sends a KM_{ADHOC} message containing MK_i. Depending on the lifetime of MK_j, N_j has three possible actions:

1. The lifetime of MK_j lies between $t_{replace}$ and $t_{no\ update}$. N_j sends a KM_{CONFLICT} message to its neighbors, containing MK_i and $kd_j=kd_i$. MK_i will propagate through the mesh and create one network.
2. If the expiry time of MK_j lies between t_{start} and $t_{replace}$, or $t_{no\ update}$ and t_{update} (see sub-sub section 5.6.2), a key update is not allowed and the meshes become

bridged. During this period, broadcast messages need to be re-encrypted and re-broadcast at either N_i or N_j.

3. Both *MK*s are the same, although generated independently. The meshes are considered to be merged. Note that the neighbors of N_j, whose *kd*-value will be higher are not affected by this change. On routing-level, routes to N_i will be created independent of KM.

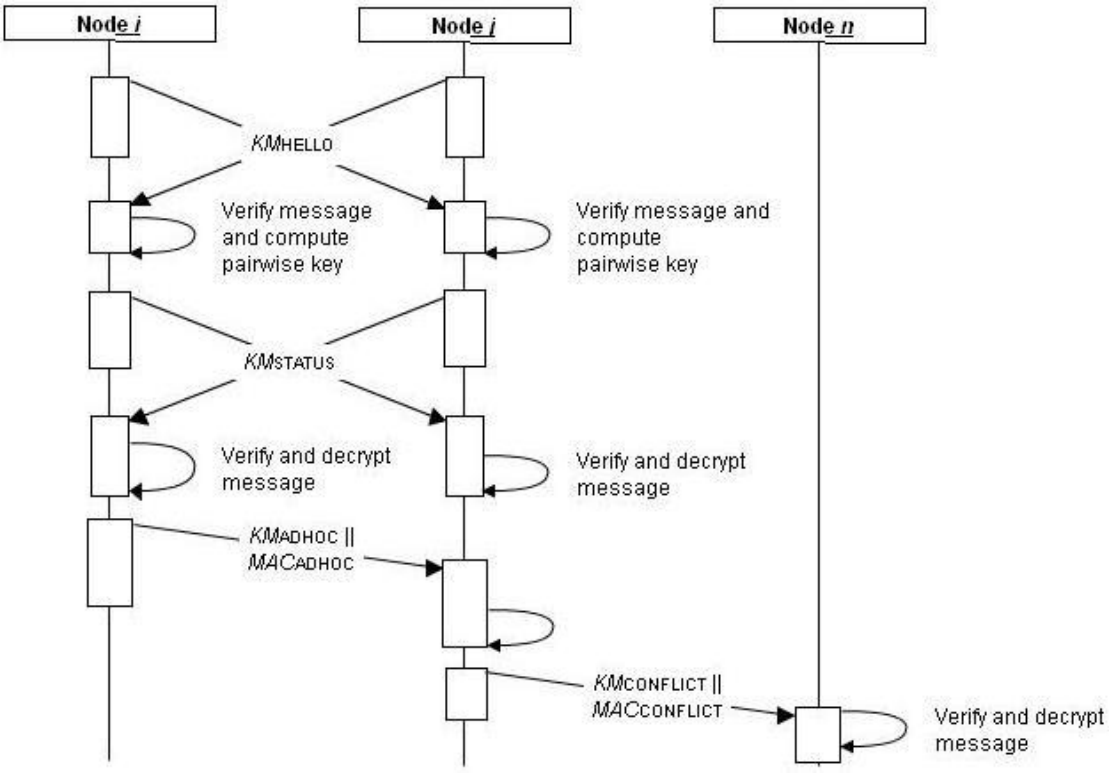

Figure 17: MSC for the mesh merging phase

Mesh merging phase:

$N_{i,j} \rightarrow *$: $KM_{\text{HELLO}} = ID_i||ID_*||\text{HELLO}||cert_i^{GC}||PK_i||(ID_i||PK_i)SK_i$

$N_{i,j}$: Verify KM_{HELLO} with PK_j,
if verification holds PROCEED,
else ABORT;

Verify $cert_j^{GC}$ with PK_{GC}
if verification holds PROCEED,
else ABORT;

Compute $K_{i,j} = hPK_jSK_i$ (see Appendix A)
where h is, PK_j is the public key from N_j
and SK_i the private key from N_i.

$N_i \rightarrow N_j$: $KM_{\text{STATUS}} = ID_j||ID_i||\text{STATUS}||NONCE_i||$
$\{NONCE_i||MAC_{i,j}||kd_i\}K_{i,j}$

$N_j \rightarrow N_i$: KM_{STATUS}

N_i : Decrypt KM_{STATUS} with pairwise key $K_{i,j}$,
if $NONCE_j$ is fresh and unaltered,
then PROCEED, else ABORT;

Check if $kd_i < kd_j$,
then use $MAC_{i,j}$ for message authentication
and PROCEED,
else STOP.

$N_i \rightarrow N_j$: $KM_{\text{ADHOC}} = ID_i||ID_j||\text{ADHOC}||NONCE_i'||\{MAC_{GROUP}||$
$NONCE_i'||MK_i||kd_i\}K_{i,j}$

$MAC_{\text{ADHOC}} = (\text{KMADHOC})MAC_{i,j}$

N_j : Use $MAC_{i,j}$ to verify MAC_{ADHOC}
if verification holds PROCEED,
else ABORT;

Decrypt KM_{ADHOC} with pairwise key $K_{j,i}$,
if $NONCE_i'$ is fresh and unaltered,
then PROCEED, else ABORT;

If $t_{replace} < t < t_{no\ update}$ PROCEED,
else ABORT.

Store MK_j;

Set $kd_j = kd_i$.

$N_j \rightarrow N_n$: $KM_{CONFLICT} = ID_j || ID_n || CONFLICT || NONCE_j' || \{NONCE_j' || (MK_i || ID_i || kd_j) K_{j,n}$

$MAC_{CONFLICT} = (KM_{CONFLICT}) MAC_{j,n}$

N_n : Use $MAC_{i,j}$ to verify $MAC_{CONFLICT}$
if verification holds PROCEED,
else ABORT;

Decrypt $KM_{CONFLICT}$ with pairwise key $K_{j,n}$,
if $NONCE_j'$ is fresh and unaltered,
then PROCEED, else ABORT;

Store MK_i;

Set $kd_n = s_j$;

Forward $KM_{CONFLICT}$ to neighbors.

To prevent replay attacks NONCEs are used during this session. Otherwise, an adversary could modify and replay the KM_{STATUS}, changing the header into "ADHOC", pretending to send an illegal KM_{ADHOC} message. N_n will distribute the key to its neighbors by sending a $KM_{CONFLICT}$ message containing MK_i, and kd_n. MK_i will propagate through the network enabling broadcasting through the entire broadcast domain.

5.4.5 Handover Phase

In KERS a handover is defined as a mobile node switching between local controllers (*LC*s). A mobile node that moves within a sub-group of the network, controlled by the same *LC*, can use one group key (*GK*). This is not considered a handover.

When N_i moves between networks, it changes the *kd*-value to $kd_i < r_{mesh}$, indicating the node has no key. N_i only establishes connections with nodes that are currently not on his node-list, to avoid connections with nodes from its former sub-group. N_i can run the node addition protocol to obtain the group key of the targeted sub-group.

5.5 Key Management Phase for Node Leaving

When a node leaves the incident site, the emergency responder must be able to indicate this manually, by pressing a button or switch on the mobile node to delete its key material. When emergency responders forget this action or an emergency vehicle gets stolen, an alternative way must be found to delete the key material.

When a mobile node has not been in contact with other mobile FIGO nodes for a certain period of time, the node must delete its key materials related to the last incident. The time period can be configured depending on the required level of confidentiality. Another alternative is when the emergency vehicle returns to its home department, a connection with the *GC* can be established. The *GC* sends a command to delete all present key material that is related to the last attended incident.

We have given an overview of KERS in the situation that connections are stable and showed that KERS covers all situations that can occur during an incident. The scheme supports a connection with the back office, but also functions in an ad hoc mesh network. It uses separate keys for different security mechanisms and the keys can be updated over the air. To show that KERS is robust for disruption, improves the security of FIGO and does not degrade performance we present a analysis that shows the robustness of the protocol, a theoretical security analysis and performance analysis in the next chapter.

6. Analysis of KERS

This chapter demonstrates the robustness for disruptions, security and performance of KERS. We have given an overview of the scheme in Chapter 5, which showed that the scheme covers all situations that can occur at an incident site. It did not take into account any disruptions of communications due to the nature of wireless connections. Additionaly, if the scheme does not satisfy the security requirements, or has performance drawbacks, KERS will not be considered a solution for FIGO.

First, we theoretically show the scheme is secure in Section 6.1. We show that KERS does not violate the security requirements for emergency response systems. Next, we show that KERS improves the security in the FIGO mesh by covering the security requirements that are not met in the security solution that is currently applied.

Section 6.2 presents a theoretical performance analysis. Although power might not be a problem for FIGO mobile nodes, there are constrains like CPU power and memory usage. The performance analysis shows that KERS can run within the resource constrains of FIGO. The performance analysis focuses on computational complexity based on cryptographic operations. It also shows the impact of the number of control messages sent into the network and how node addition and revocation affects the network.

This chapter concludes with a discussion on implementation issues in Section 6.3. In this section we show that KERS is robust for disruption of communications and that few, simple, alternative protocols ensure that KERS is robust. Because FIGO is a product in development patents and licensing has to be taken into account. This section serves to bridge the gap between the theoretical solution without restrictions, and the practical limitations that any academic proposal faces.

6.1 Security Analysis

To show that KERS satisfies the security requirements for emergency response systems we conduct a theoretical security analysis. In this analysis we first, briefly

recall the security requirements from Section 2.1. Then, we show that KERS does not violate any of the security requirements. We show in particular that KERS improves the security of FIGO by satisfying the security requirements that are not met with current the key management scheme.

Any device that participates in the emergency response system must authenticate itself to other legitimate devices (R1). Messages must only be readable by the parties sharing the channel (R2). Access to the emergency response network is restricted to authorized nodes belonging to the same emergency response service (R3 and R4). Confidential data can only be decrypted by legitimate nodes (R5). Control traffic must be reduced to a minimum to maintain optimal performance (R6). Besides communicating within an ad hoc mesh network, emergency response systems must also be able to communicate with the back office, if the communication infrastructure allows this (R7). Message integrity must be assured to prevent (un)accidental data alterations and message authentication must be included to help preserve data integrity (R8 and R9). Key management schemes must support over the air re-keying and use separate keys for different mechanisms (R10 and R11). Nodes must not deny having sent or received messages (R12). The emergency response systems and its mechanisms must scale well to support node mobility and network dynamics (R13). With regard to service availability, it is important that the emergency response system is self-healing (R14 and R15).

Device authentication in KERS is established by certificates. Some instances of certificates (for example PGP[1]) can be signed by different organizations to prove the node is legitimate. In our scheme, we propose that at least two parties sign the certificate from a node. First, the certificate must be signed by the production company to prove that legitimate manufacturers produced the node is. When the node

[1] http://www.pgpi.org

can prove it comes from a trustworthy manufacturer, the group controller (*GC*) can sign the certificate to bind the node to a specific emergency department. Additionally, the node receives a separate certificate chain that specifies other important groups that the node belongs to, e.g. geographical related groups or emergency service related groups. Based on the certificate, nodes establish unicast and broadcast channels. To transfer the keys for secure communications over those channels, nodes use their public and private key. The keys are related to the certificate. Based on the security of the certificate, a node is certain that the channels it shares with other nodes are only shared by legitimate parties. KERS does satisfy the authentication requirements R1 and R2.

Authorization and access control in KERS are enforced during the pre-deployment state. When a node does not go through the pre-deployment phases, it is unable to participate in the KERS protocol. Only legitimate, registered nodes can access the emergency response network. Second, nodes can only communicate with other nodes that belong to the same *GC*. Even a stricter network access policy is enforced in KERS by letting each local controller (*LC*) have a unique device secret. Broadcast communication is therefore restricted to nodes belonging to the same *LC*, while unicast communication is possible between nodes belonging to the same *GC*. KERS satisfies R3 and R4.

Confidential data can be encrypted in KERS to prevent adversaries from eavesdropping. The unicast key cannot be intercepted, since it is computed locally based on parameters that are installed during the production phase, which is considered secure and it is assumed that neither insider nor outsider can acquire these parameters from the device itself. The group key (*GK*) is computed by *LC*, while the mesh key (*MK*) is generated by a mobile node. When *LC* computes *GK* it uses its unique device secret and encrypts the key using the pairwise key. Adversaries are unable to intercept and decrypt this message, and thus unable to acquire *GK*. When a

mobile node generates *MK*, it distributes the key encrypted with the pairwise key, which implies that the adversary cannot decrypt the message. Adversaries are unable to acquire *MK*. As an alternative, adversaries can try to put their own keys in the network, but since only keys that are encrypted with a pairwise key are accepted, this is infeasible. Adversaries that want to establish a pairwise key with a legitimate node must have been through the production phase. KERS satisfies security requirement R5.

In order to minimize control traffic, KERS specifies a protocol that limits the periodic broadcast of KM_{HELLO} messages. Since no practical implementation has been realized, no simulations results are available to ensure whether control traffic is minimized. We can therefore not make any serious statements with regard to R6. KERS supports mobile nodes that want to establish a connection with the back office, satisfying R7.

To detect (un)intentional data alterations, KERS specifies two mechanisms. First, most key management messages include NONCEs to detect if encrypted messages have been altered. Additionally, KERS uses a message authentication code over the whole message. By using standardized methods and algorithms, KERS satisfies R8 and R9.

All credentials in KERS can be updated during an incident, except the pairwise key, which is based on the public-private key pair that is also not updated during an incident. R10 specifies that all keys must be updatable during an incident and it seems that KERS violates R10. The pairwise key can in theory be updated, because nodes can generate a new public-private ECDH key pair at any point in time. However, this is a costly operation. First, the generation of the key pair takes more time than generating a new symmetric key. Second, the distribution of PK_i would require a KM_{HELLO} message, in the current design, and sending this message is restricted by a protocol to limit the overhead generated by KERS. The re-distribution of the public

key could thus change the performance of KERS. Finally, the pairwise key must be recomputed, which is an expensive ECDH computation (see Section 6.2). Due to these three reasons we decided not to include an update mechanism for the pairwise key. This results in a partial violation of R10, although this can be corrected by redefining the re-keying protocol and take the performance drawbacks into account. KERS uses separate keys for confidentiality, integrity, device and channel authentication and message authentication. This results in a partial violation of R10 while satisfying R11.

KERS does not explicitly specify an explicit logging functionality that could provide non-repudiation, but it does not conflict with the scheme. Both unicast and broadcast messages are sent with key material that is shared by multiple nodes and thus encryption keys do not provide non-repudiation. When a message authentication code is used, the sender can be traced based on the initialization vector, leading to non-repudiation for the sender. The receiver however, can always claim not having received the message. We conclude that KERS does not satisfy R12.

By appointing *LC*s the FIGO mesh is divided in different sub-groups. This division ensures that node revocation does not affect the whole network. The number of sub-groups is arbitrary and prevents that any part of the network becomes too large to maintain, leading to sub-optimal performance or security. The possibility to merge or bridge different networks ensures that there is no restriction to the number of nodes that can join the emergency response network. KERS does satisfy security requirement R13.

To ensure that the disruptions to the emergency response network do not make the network unavailable permanently, KERS takes two precautions. First, nodes that are disconnected and re-connect before the current key expires do not need to re-authenticate. Second, when the connection with the back office is lost, local controllers will generate a new *GK* when the current *GK* expires. This is less

expensive than switching to the ad-hoc distribution protocol that generates a *MK*. KERS satisfies R14 and R15.

KERS improves the security of FIGO by satisfying R2, R8, R9, R10 partially, R11, R13 and R15. To fully satisfy R10 we need to redefine the re-keying protocol and revise our performance analysis. For non-repudiation KERS does not explicitly improve the security of FIGO. Non-repudiation can as easily be added to FIGO as it must be added to KERS. In conclusion, we state that KERS does significantly improve the security of FIGO. In the next section, we discuss the performance of KERS to show that besides an improvement in security, the scheme does not degrade the performance of FIGO.

6.2 Performance Analysis of KERS

This section provides a theoretical performance analysis of the KERS protocol. The protocol is evaluated based on the number of asymmetric and symmetric computations. Based on the literature, an educated guess is made about the time complexity and resource limits, which must be acceptable for FIGO. The results should not reveal any performance penalties, which would make the protocol unfit for FIGO.

To evaluate the number of computations of KERS we count the number of symmetric (AES) and asymmetric (ECC) computations. Performance in KERS cannot be analyzed by evaluating the number of computations for one node, since not every protocol in KERS is symmetric. We therefore distinguish between the computations made by N_i and N_j. Not every state includes communication asymmetry between N_i and N_j, leading to a shorter overview in computations for N_j. The combined total of computations by N_i and N_j is the performance measure of KERS.

The total number of cryptographic computations in KERS for N_i is shown in the following table. We do not distinguish between encryption, decryption, signing and verification yet.

Table 6: Summary of cryptographic computations in KERS made by N_i

State	# Symmetric computations	# ECC computations
Pre-deployment	6	2
Neighbor discovery between two nodes	2	4
Neighbor discovery with *GC*	1	2
GC involvement	6	-
No *GC* involvement	11	6
Total	24	14

When we split the ECC operations in signing, verifying and key generation we get the following table for N_i.

Table 7: Specific ECC operations in KERS for N_i

Phase	ECC signing	ECC verifying	ECDH key generation
Registration phase	-	1	1
Neighbor discovery with *GC*	1	1	-
Neighbor discovery between mobile nodes	1	2	1
Node addition	1	-	1
Mesh merging	1	2	1

N_j does not communicate with *Ni* during the pre-deployment state. *WEB* and *GC* that do communicate with *Ni* are considered to be devices that are not resource constrained. The neighbor discovery protocol is one of the few symmetric protocols, and therefore it is not repeated here for N_j. The remaining states of KERS where N_j does have to perform additional computations are listed in the table below.

Table 8: Summary of additional cryptographic computations in KERS made by N_j

Phase	# Symmetric computations	# ECC computations
GC involvement	9	-
No GC involvement	17	7

Table 9: Specific ECC operations in KERS for N_j

Phase	ECC signing	ECC verifying	ECDH key generation
Node addition	-	2	1
Mesh merging phase	1	2	1

From the total number of ECC computations we conclude that the number of ECC computations is relatively small, compared to the number of AES computations. Assuming that the computation time of AES computations is negligible compared to public key computations we find that the ratio between AES and ECC computations is acceptable.

To evaluate the performance of ECC on mobile nodes we compare the relevant hardware of characteristics of mobile FIGO nodes with the hardware of the MSP 430 controller, which is briefly described in [19]. A study from Gouvea and Lopéz [19] shows that ECC has no performance penalties on the MSP 430 controller. We give an estimation of the time complexity for ECC computations on a mobile FIGO node.

A mobile FIGO node uses an Intel XEON 500 MHz processor on an Intel 5100 Chipset. The instruction set of the MSP 430 does not support multiply and divide, which implies that multiplication and division operations create overhead. Since FIGO hardware does support these operations directly, the MSP 430 provides an even better benchmark. Additionally, the research described in [19] uses an 8 MHz processor.

The researchers use ECDSA as ECC implementation at 80-bit level security using the secg160r1 curve [13]. For ECDSA on the MSP 430 controller signing operation take 2,166,906 clock cycles, taking 270 ms. Verification is the most costly operation taking 5,488,568 clock cycles and 686 ms. The memory allocation varies from 25.7 KB to 31.3 KB ROM and from 3.5 KB to 2.9 KB RAM depending on the version used.

Since ECC computations in KERS are, sign/verify and ECDH key generation operations we also refer to TinyECC [29] where simulation results are presented based on ECDH. The research is performed on multiple processors, and we take as benchmark the results on a comparable system: the Tmote Sky on 8 MHz. Since time efficiency is more important than memory efficiency, we focus on the least time consuming optimization. ECDH initialization takes 179.16 ms. and key establishment takes 392.12 ms.. We take the total time of 571.28 ms. into account during ECDH key generation.

Based on these numbers we create a table where we detail the time for the ECC computations in each state, for Ni and N_j separately. Since we did not use ECDSA in KERS, the comparison with [19] is not highly reliable. However, looking at the differences between ECDSA and ECDH in TinyECC, we note that these are very small in timing as well as memory usage [29]. Additionally, the ECDSA computations use a benchmark system with less resources and computational power than a mobile FIGO node. Therefore, we do use the two references as performance measure.

Table 10: ECC timings for N_i, based on MSP 430 8MHz

State	Signing time (ms)	Verification time (ms)	ECDH Key generation time (ms)	Total time (ms) spent on ECC computations
Registration phase	-	686	571.28	1257.28
Neighbor discovery with GC	270	686	-	956
Neighbor discovery with two mobile nodes	270	1372	571.28	2213.28
Node addition	270	-	571.28	841.28
Mesh merging	270	1372	571.28	2213.28

Table 11: ECC timings for N_j, based on MSP 430 8MHz

State	Signing time (ms)	Verification time (ms)	ECDH Key generation time (ms)	Total time (ms) spent on ECC computations
Neighbor discovery with two mobile nodes	270	1372	571.28	2213.28
Node addition	-	1372	571.28	1943.28
Mesh merging phase	270	1372	571.28	2213.28

From this table we infer that the total time to perform node addition, merge two mesh networks and perform neighbor discovery between two nodes takes much time, around 2-4 seconds. However, we note that the mobile FIGO nodes have much better hardware. From the TinyECC experiments, we see that much better results are gained from a 416 MHz controller. When we take these results as benchmark, the results are as follows.

Table 12: ECC timings for *Ni*, based on Imote2 416MHz

State	Signing time (ms)	Verification time (ms)	ECDH Key generation time (ms)	Total time (ms) spent on ECC computations
Registration phase	-	14.49	18.54	33.03
Neighbor discovery with GC	11.80	14.49	-	26.29
Neighbor discovery with two mobile nodes	11.80	28.98	18.54	59.32
Node addition	11.80	-	18.54	30.34
Mesh merging	11.80	28.98	18.54	59.32

Table 13: ECC timings for N_j, based on Imote2 416MHz

State	Signing time (ms)	Verification time (ms)	ECDH Key generation time (ms)	Total time (ms) spent on ECC computations
Neighbor discovery with two mobile nodes	11.80	28.98	18.54	59.32
Node addition	-	28.98	18.54	47.52
Mesh merging phase	11.80	28.98	18.54	59.32

From these results, we see that the neighbor discovery phase, mesh merging and node addition phase take no longer than 80-120 ms, which is acceptable. We state that based on our theoretical analysis the mesh merging and node addition phase take no longer than an acceptable 0.5-0.75 seconds in ECC computations.

In this section, we showed that KERS is a suitable candidate for FIGO based on a theoretical performance analysis comparing the mobile FIGO nodes with slower

hardware. Besides performance measures, KERS has to deal with other practical implementations. In the next section, we show how such issues might have implications for KERS, when improving the security of FIGO.

6.3 Robustness for Disruptions of Wireless Connections

In Chapter 5, we presented how KERS works when connections remain stable during the execution of the protocols. However, in wireless environments we cannot make this assumption. Since KERS is designed for emergency response systems that operate in wireless environments, it is important that the scheme handles disrupted protocols as well.

When considering disruption of wireless connections, we only have to research the impact on protocols that take place in wireless environments. It is not specified whether the production phase uses place wireless communication, but to minimize security threats it is likely that is does not. The production phase is therefore not considered here.

The *registration phase* makes use of wireless connections. It involves three entities and disruptions may cause each entity to have a different view on the progress of the protocol. It makes it hard to restart the protocol, because signing a certificate more than once may result in a security breach. To minimize the impact of disruptions, the connection between group controller (*GC*) and the web interface (*WEB*) must not only be secure but also be reliable. If we assume that both entities are fixed entities with enough resources to run TCP, we can state that underlying mechanisms ensure that the KM_{SIGN} and KM_{SIGNED} message are received properly. By this assumption, we ensure that certificates are only signed once. When the KM_{REGISTER} message gets lost, N_i will transmit it again after a time-out occurs, see Figure 18. In case the $KM_{\text{REGISTERED}}$ message gets lost, the protocol will have to re-run after a time-out, the alternative protocol excludes the communication between the *WEB* and the *GC*, shown in Figure 19. A problem occurs when adversaries replay an earlier sent

KM_{REGISTER} message to acquire the encrypted certificate to perform offline attacks on it. We are aware that interception of the $KM_{\text{REGISTERED}}$ message is always possible. Although encryption mechanisms are in place to minimize the chance of illegitimate use of a certificate, the lifetime of a certificate must be chosen to abuse the vulnerability in the registration protocol.

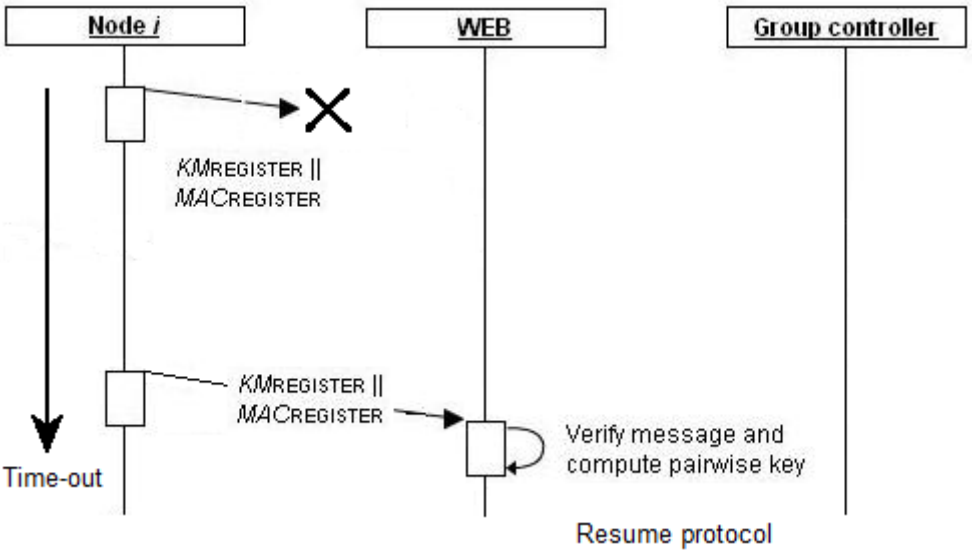

Figure 18: Alternative registration protocol when the first message is disrupted

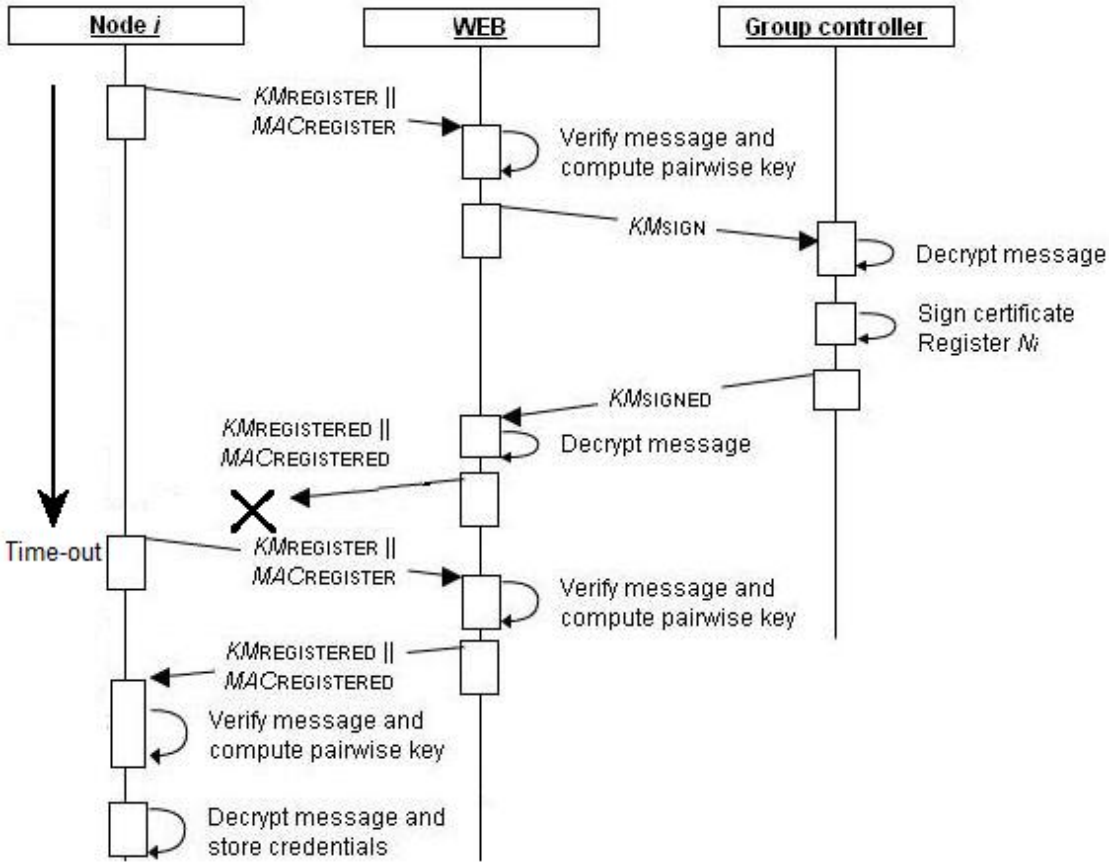

Figure 19: Alternative registration protocol when the $KM_{\text{REGISTERED}}$ message is disrupted

The *bootstrap protocol* is similar to the registration protocol in terms of entities and message exchange. The impact of disruptions is similar, because of the required confidentiality for bootstrap parameters. The alternative protocol that includes a time-out and re-transmission is therefore similar to that of the registration protocol and we omit the alternative MSCs.

The *neighbor-discovery protocol* consists of the exchange of four messages between two entities. Because KM_{HELLO} messages are broadcasted until the node has received a key, the impact of disruption during KM_{HELLO} transmissions is minimal. The exchange of KM_{STATUS} messages determines the future actions of a node. It is important that both nodes receive each other's KM_{STATUS} message to acquire knowledge of the current state of the network. Implicitly KERS covers one case of

disruption of the neighbor-discovery protocol. When N_j has acquired a key (either a *GK* or *MK*) from a different node during the disruption of the KM_{STATUS} message and N_i has not received the KM_{STATUS} message from N_j, N_i will continue to broadcast KM_{HELLO} messages indicating it is in the "No key"-state. When N_j notices this message, both nodes will engage in the *node addition protocol* see Figure 20. When both KM_{STATUS} messages are lost because of disruption, the registration protocol will re-run. Therefore, the neighbor discovery protocol is robust for disruptions.

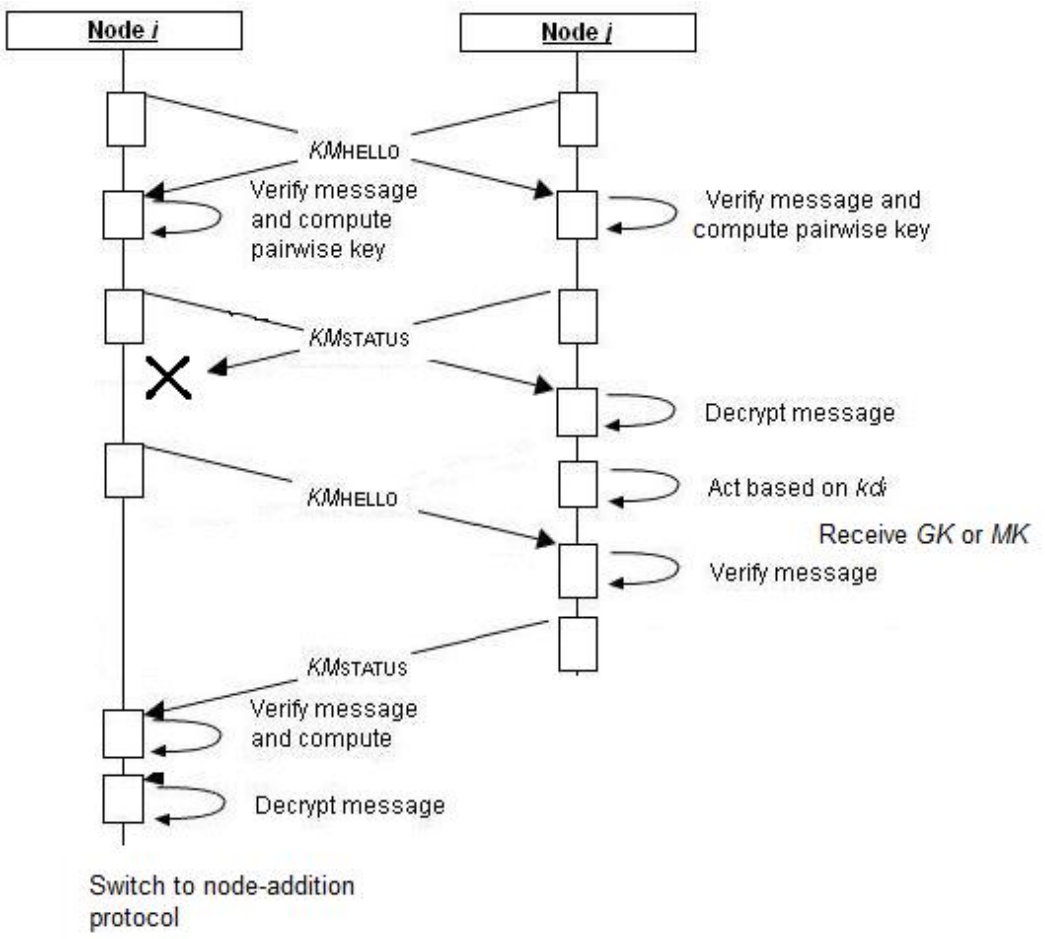

Figure 20: Alternative neighbor discovery protocol for two nodes

Disruptions that occur during *neighbor discovery* involving the group controller (*GC*) and N_i have a smaller impact than neighbor discovery between two mobile nodes. The KM_{HELLO} message from N_i is broadcasted periodically until it receives a

key. Suppose the message from the *GC* that is send to N_i is lost, then N_i will simply think the *GC* is offline. When this happens with all interactions between the *GC* and any mobile node, the mobile nodes will form an ad-hoc mesh network, without the *GC*. If the *GC* manages to establish a connection with one mobile node, that mobile node will become a local controller (*LC*) and computes and distributes the group key (*GK*). Disruptions do not affect the neighbor-discovery protocol such that it degrades the performance of KERS. MSCs for these situations are omitted.

In our network model, each FIGO mesh that is connected to a *GC* has one *LC*. When the connection between the *GC* and *LC* is lost during the *group key distribution protocol*, group members (*GMs*) will keep waiting for the *LC* to distribute the group key (*GK*). Until the *GK* is distributed, the network will not perform. To avoid this scenario the *LC* can follow a time-out mechanism and when the time-out is reached, the FIGO mesh can either run the *ad-hoc distribution protocol* or the *LC* can generate a mesh key and distribute it to the *GMs*, pretending it to be a *GK*, as is shown in Figure 21. This last option is also specified in the *re-keying mechanism* of the *GK*, which itself is robust for disruptions.

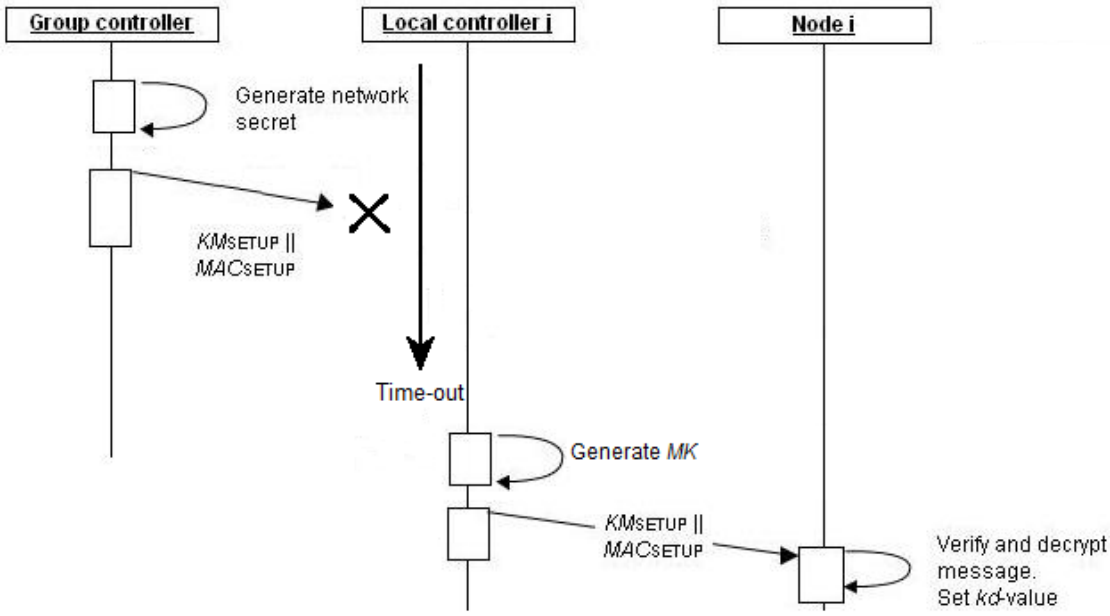

Figure 21: Alternative group key distribution protocol

The *network interoperability protocol* resembles the group key distribution protocol, with the difference that the *LC* cannot generate an interoperability key. When the connection between the *GC* and *LC* is lost, the protocol cannot take place. Network interoperability is coordinated from the back office, and disruption of the protocol can be communicated to the other back office. The network operability protocol can be seen as an atomic action that takes occurs as a complete single protocol run, or does not occur at all. Therefore we do not model an alternative and MSC.

Decentralized key distribution depends on multiple transmissions containing the mesh key (*MK*). The protocol for two nodes contains one transmission and when considering disruptions, two relevant scenarios can occur. The first is when N_j, the node that is responsible for generating *MK*, gets disconnected to all its neighbors. In this case the protocol will re-run. When the communication between N_i, a neighboring node of N_j, and N_j gets disrupted, N_i will not receive *MK*. If N_i can acquire *MK* from a different neighbor, it will receive the *MK*. If this cannot occur, N_i will have to wait until the connection with N_j is restored. The impact on the performance will be minimal and an alternative protocol is not needed.

The *node addition protocol* has four message exchange operations that can be disrupted. Again, disruption of the KM_{HELLO} message has no impact on the protocol. When transmission of the KM_{STATUS} messages is disrupted, the impact is minimal because the node to be added does not change its *kd*-value and continues broadcasting KM_{HELLO} messages. When the joining node does not receive the KM_{SETUP} message after a certain period of time it can resume broadcasting KM_{HELLO} messages. When the sender of the KM_{SETUP} message receives this message, it knows the KM_{SETUP} was not delivered. The protocol can switch to a *distribution protocol* because only the transmission of a KM_{SETUP} is needed, as shown in Figure 22. Otherwise, the joining node finds a different node to run the node addition protocol with.

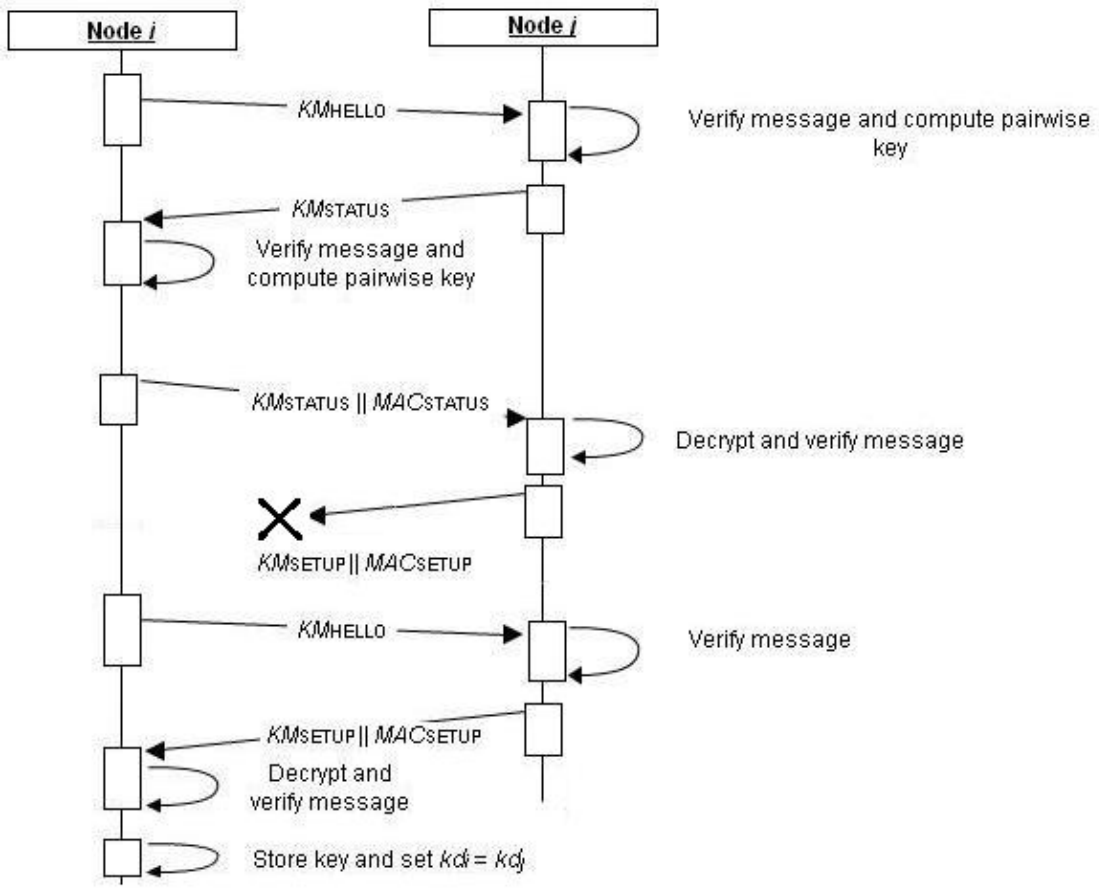

Figure 22: Alternative node addition protocol

The impact of disruptions on the *mesh merging protocol* is expected to be minimal. The worst consequence is that the meshes will not merge, but performance will not suffer from this consequence. Disruptions of the KM_{HELLO} messages have little impact on the protocol. Such disruptions will cause the protocol to restart. The exchange of KM_{STATUS} messages can be disrupted in three ways. *First*, both messages can get lost. In that case the protocol will have to re-run. *Second*, only the KM_{STATUS} message coming from N_j can get lost. When $kd_j < kd_i$ the disruption will have minimal impact, because N_i notices it has the lowest *kd*-value and must send the KM_{ADHOC} message. Because kd_i is also sent within the KM_{ADHOC} message, N_j can update its *kd*-value accordingly. When N_j receives this message, it will not know that its KM_{STATUS} message got lost. In the *third* case, as shown in Figure 23, the KM_{STATUS}

message from N_i is lost while $kd_j < kd_i$. N_j waits for N_i to send the KM_{ADHOC} message, while N_i waits for the KM_{STATUS} message from N_j. Because N_j will periodically send its *kd*-value, the protocol can continue if N_i receives the KM_{STATUS} message within an acceptable period of time. In this case, it can be considered a re-run of the protocol. Finally, forwarding the new *MK* to neighbors of N_j will not be affected since KM_{STATUS} message are periodically sent and N_j will notice if the transmission was disrupted.

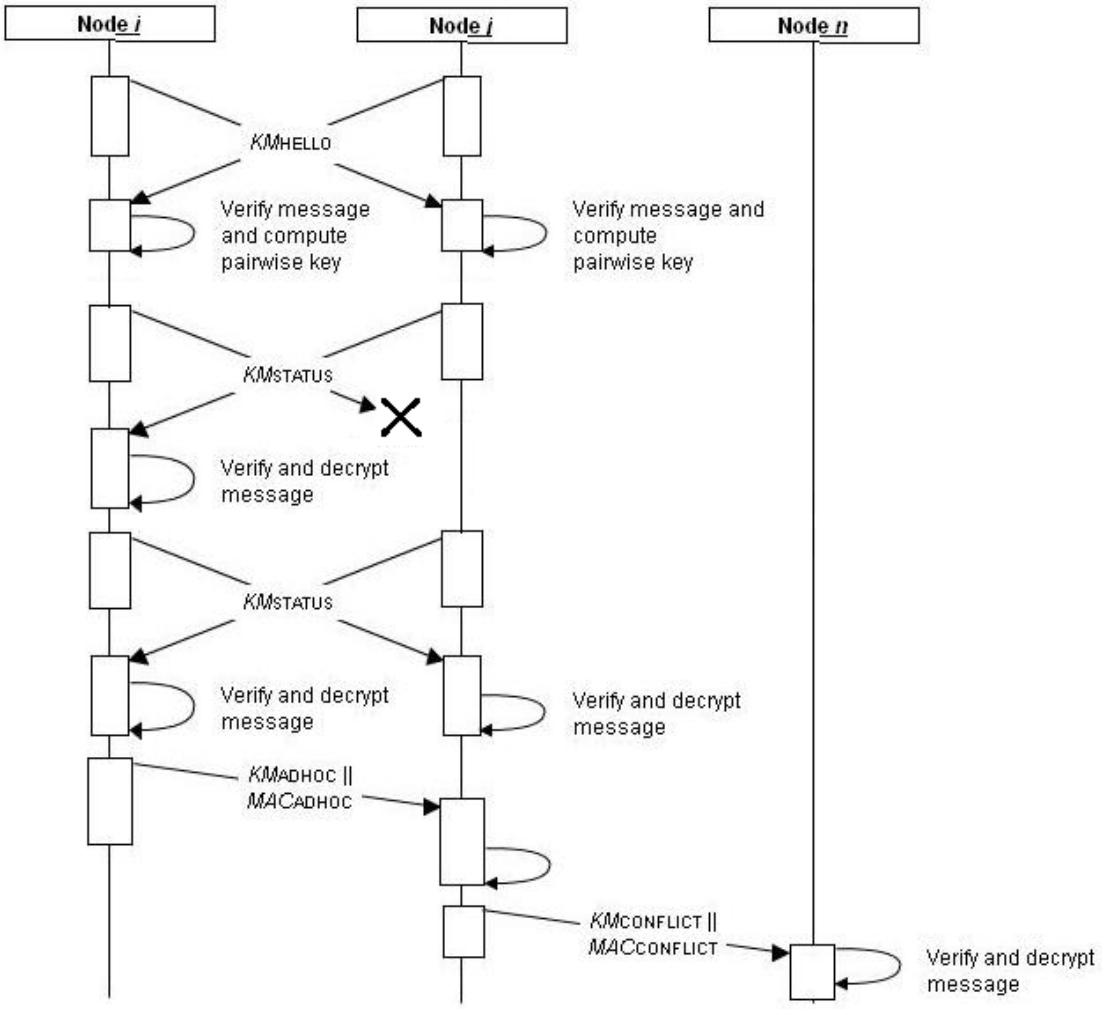

Figure 23: Alternative mesh merging protocol

We have shown that KERS is robust for disruptions of wireless communications. Most protocols can use a time-out mechanism to re-transmit a message that indicates

the loss of a transmission. In some cases protocols do not have to re-run and can switch to other protocols or resume by re-sending the lost message. A few cases require the protocol to re-run. This might add to the number of encryptions and decryptions and affect performance; this should be taken into account when performing practical simulations.

6.4 Practical Implementation Issues of KERS

Besides theoretical limitations, KERS must deal with practical issues. In this section, we discuss three of the most important practical implementation issues. The issues show that although KERS is a satisfactory solution for emergency response systems, it has drawbacks that must be considered before implementing the scheme.

KERS makes use of elliptic curve cryptography (ECC) because of its suitability in mobile environments. However, there are some practical implications when using ECC. From its early introduction Certicom has patented over 100 ECC and public key cryptography techniques [36]. This has lead to a delayed adoption of ECC into the public domain. As a consequence, companies are reluctant to implement an open source version of ECC into their systems. Sony was filed a lawsuit in 2007 by Certicom, but the case was dismissed[2]. However, for a company producing FIGO such a risk is unacceptable. Licensing an ECC implementation would be a good alternative, however the costs for such a solution might be higher than the company is willing to pay.

When ECC is not used in KERS, there must be a similar alternative for public key cryptography to maintain the scheme its security and efficiency. Alternatives are RSA and DLP based cryptographic techniques like ElGamal [9], DSA [33] and DHIES [1]. There exist proposals using RSA in wireless environments, e.g. improving the

[2] http://docs.justia.com/cases/federal/district-courts/texas/txedce/2:2007cv00216/103383/113/

WiMAX standard with RSA or ECC [21]. The same holds for DHIES, which can be optimized for wireless networks [4].

When these alternatives provide unsatisfactory performance, the final alternative is to remove the public key cryptography and replace it with symmetric cryptography. Using only symmetric cryptography will substantially weaken the scheme, since device and message authentication are harder to achieve. Additionally, it becomes harder to bootstrap nodes with pairwise keys, resulting in a shift towards pre-shared keys without over the air re-keying.

Finally, we discuss an open issue regarding the overhead that is generated by KERS. In wireless environments it is important to estimate the number of bits in the air, since this determines the general performance of the network. Wireless environments are suspect to collision, and the number of collisions increases with the number of bits that are transported through the air. Unfortunately due to the lack of a practical implementation and simulation results, we cannot state any solid argument about the expected number of bits in the air that are due to KERS. We can only state that we have tried to limit the number of transmitted bits by presenting a protocol that specifies the frequency that KM_{HELLO} messages are sent. This frequency is not a fixed number and can be adjusted to optimize the ratio between the time it takes for nodes to acquire the group key and the number of bits in the air. It holds that when the number of bits in the air is minimized it will maximize the time it takes for nodes to establish neighbor discovery. We expect this to be a linear relationship with negative correlation, and do not think there is an optimum like in a hyperbolic relationship. However, these hypotheses need empiric evidence from simulations.

7. Conclusion and Future Work

The goal of this thesis was to give an overview of the security architecture of FIGO, evaluate the security architecture and propose an improvement to the security architecture of FIGO. Currently the security architecture of FIGO is not well-documented. Neither is the security architecture evaluated to meet the security requirements. More importantly, the FIGO security architecture does not borrow ideas from similar systems that function as potential benchmarks for FIGO. Extending the FIGO security requirements with security requirements of the MESA and SafeCom project have given insight into the potential weaknesses of FIGO. The evaluation of the FIGO security architecture concludes that key management in FIGO is a weak spot. Over the air re-keying (OTAR) is impossible in FIGO and data integrity does not meet general criteria, using a CRC instead of a widely accepted method. The weaknesses lead to the endangerment of data confidentiality and data integrity as well as device authentication.

This thesis proposed KERS to improve key management in FIGO. KERS supports OTAR and suggests the use of a secure hash function to protect integrity. Theoretical security analysis shows that KERS meets the security requirements for emergency response systems. KERS does add to the chance of successful DoS attacks, although such attacks can never be prevented. The main drawback of this thesis is that it lacks a formal theoretical and practical performance analysis. The current theoretical performance analysis shows that KERS can be implemented in FIGO without performance penalties, improving the current security architecture. However, there is no practical evidence that the protocol will recover from the potential DoS attacks. Another drawback is that this thesis does not take the impact on human policy into account. KERS makes the assumption that underlying human policies are trustworthy and any additions to such policies because of KERS are not considered. It is recommended that a usability analysis be performed based on the human actions that are required in KERS. Such an analysis must take into account whether the

administrator of an emergency service department gets an increased responsibility with regard to security issues. In addition, the responsibility of emergency responders that operate the emergency vehicles needs to be analyzed. Finally, an assessment must be made that analyzes if the change in responsibilities has an impact on the current human policies that are drafted for emergency responders and administrators. Based on this assessment conclusions about the human performance side of KERS can be drafted, which add to the total performance of KERS.

In the near future FIGO will add a low-cost mobile node to the production line. The node will have less expensive hardware, less computational power and possibly run on batteries. KERS has not considered such energy constraints. However, practical studies have shown that performance of the cryptographic components of KERS is acceptable in low-cost environments. Considering that cryptographic computations are the performance bottleneck in key management, KERS is applicable on the low-cost mobile node. The implications of this finding are important, because it would be infeasible to employ two different key management protocols on FIGO nodes. In the future, it is recommended to conduct experiments with different implementations of the KERS protocol and do a performance analysis. The performance analysis can be performed on the regular FIGO node as well as the low-cost variant. When performance is similar, the KERS protocol can be adopted as uniform solution for the FIGO project.

References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. *In David Naccache, editor, CT-RSA 2001*, Vol. 2020 of LCNS, pp. 143-158, April 2001.

[2] W. A. Arbaugh. An inductive chosen plaintext attack against WEP/WEP2. *IEEE Document 802.11-01/230*, May 2001.

[3] W. A. Arbaugh, N. Shankar, and Y. J. Wan. Your 802.11 wireless network has no clothes. *http://www.cs.umd.edu/~waa/wireless.pdf*, Mar. 2001.

[4] J. Baek, H. C. Tan, J. Zhou, and J.W. Wong. Realizing stateful public key encryption in wireless sensor networks. *In Proceedings of The IFIP TC-1123rd International Information Security Conference (SEC '08), pp. 95-107*. 2008

[5] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: The insecurity of 802.11. *MOBICOM 2001* (2001).

[6] M. Bouassida, I. Chrisment, and O. Festor. Efficient clustering for multicast key distribution in MANETs. *LCNS* 3462, pp 138-153, May 2005.

[7] M. Bouassida, I. Chrisment, and O. Festor. Group key management in MANETs. *International Journal of Network Security*, pp. 67-79, Jan. 2008.

[8] F. Demertzis, C. Xenakis. SOMA: Self-organized mesh authentication. In *Proc. 7th European Workshop on Public Key Services, Applications and Infrastructures (EuroPKI´10)*, In Press, Sept. 2010.

[9] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, Vol. 31, pp. 469-472, 1985.

[10] M. Bellare, R. Canetti, and H. Krawczyk. Keyed hash functions and message authentication. In *Proceedings of Crypto'96, LNCS 1109*, pp. 1-15, 1996.

[11] M. Bellare, R. Canetti, and H. Krawczyk. HMAC: Keyed-hashing for message authentication, *IETF Networking Group, RFC2104*, 1997

[12] Certicom Research, *SEC 1: Elliptic Curve Cryptography*, 1st edition, Sept. 2000.

[13] Certicom Research, *SEC 2: Recommended Elliptic Curve Domain Parameters*, http://www.secg.org, 2006.

[14] Y. Challall, H. Bettahar and A. Bouabdallah. A scalable and adaptive key management protocol for group communication. *Wired and Wireless Internet Communications (WWIC'04*), *LCNS*(2957):260-271, February 2004.

[15] Department of Homeland Security. Public safety statement of requirements for communiactions and interoperability. Oct. 2006 [online]. Available at http://www.safecomprogram.gov/NR/rdonlyres/986E1584-1670-4BFE-9F0B-FA990C886D90/0/SoR1_v12_03122008.pdf, Accessed on 20 July 2010.

[16] W. Diffie and M. Hellman. New directions in Cryptography. *IEEE Transactions on Information Theory*, pp. 644-654, 1976.

[17] L. R. Dondeti, S. Mukherjee, and A. Samal, Scalable secure one-to-many group communication using dual encryption. In *Computer Communications, vol. 23, no. 17*, pp. 1681-1701, 2000.

[18] J. Douceur. The Sybil attack. *Proc. of the IPTPS Workshop*, pp. 251-260, March 2002.

[19]C. P. L. Gouvêa and J. C. Lopez. Software implementation of pairing-based cryptography on sensor networks using the MSP430 microcontroller. *In Cryptology INDOCRYPT 2009, volume 5922 of Lecture Notes in Computer Science*, pages 248-262, 2009.

[20] V. Gupta, S. Krishnamurthy and M. Faloutos. Denial of service attacks at the MAC layer in wireless ad hoc networks. *Proc. of 2002 MILCOM Conference*, pp. 1118-1123, Oct. 2002

[21] M. Habib, T. Mehmood, F. Ullah, and M. Ibrahim. Performance of WiMAX security algorithm: the comparative study of RSA encryption algorihm with ECC encryption algorithm. *In Proceedings of the International Conference on Computer Techlogy (ICCTD '09)*, pp. 108-112, Nov. 2009.

[22] C. He, J.C. Mitchell. Security analysis and improvements for IEEE 802.11i. *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS'05)*, pp. 90– 110, 2005.

[23] W. He, Y. Huang, K. Nahrstedt and W.C. Lee. SMOCK: A self-contained public key management scheme for mission critical wireless ad hoc networks. In *PERCOM '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications*, pp. 201–210, 2007.

[24] R. Housley, W. Polk, W. Ford and D. Solo. RFC 3280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *Network Working Group - Request for Comments*, 2002.

[25] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11, 1999 Edition*, 1999.

[26] IEEE P802.11i/D10.0. Medium Access Control (MAC) Security Enhancements, Amendment 6 to IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – *Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. April, 2004.

[27] IEEE8021X. Port-based Network Access Control. *IEEE Std 802.1x, 2001 Edition. IEEE Standard*, June 2001.

[28] L. Law, A. J. Menezes, M. Qu, J. Solinas, S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, pp.119-134, 2003.

[29] A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless networks. *7th Conference on Information Processing on Sensor Networks (IPSN 2008)*, pp. 245-256, 2008.

[30] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[31] S. Mittra. Iolus: A framework for scalable secure multicast. In *Proceedings of ACM SIGCOMM'97*, September 1997.

[32] NIST. Secure hash standard. Federal Information Processing Standard, FIPS-180-1, April 1995.

[33] NIST. Digital Signature Standard. Federal Information Processing Standard, FIPS-186-1, May 1994.

[34] NIST. Advanced Encryption Standard: Federal Information Processing Standard, FIPS-197, Nov. 2001.

[35] NIST. The keyed-hash message authenctication code: Federal Information Processing Standard, FIPS-198-1, Jul. 2008.

[36] National Security Agency. *The Case for Elliptic Curve Cryptography*, Jan. 2009. [Online]. Available at: http://www.nsa.gov/business/programs/elliptic_curve.shtml, Accessed 25 January 2010.

[37] E. A. Panaousis and C. Politis. Securing ad-hoc networks in extreme emergency cases. In *World Wireless Research Forum Meeting*, 2009.

[38] Project MESA. Statements of requirements. Jan. 2005 [online]. Available at http://www.projectmesa.org/ftp/Specifications/MESA_70.001_V3.1.2_SoR.doc, Accessed 13 July 2010.

[39] C. Rigney, S. Willens, A. Rubens, and W. Sympson. Remote Authentication Dial In User Service (RADIUS). *RFC 2865*, June 2000.

[40] M.S. Siddiqui and C.S. Hong. Security issues in wireless mesh networks. *Proc. of IEEE Int. Conf. on Multimedia and Ubiquitous Engineering (MUE) 2007*, pp. 717-722, April 2007

[41] D. Simon, B. Aboba, and R. Hurst. The EAP-TLS Authentication Protocol. *RFC 5216, IETF*, 2008.

[42] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A new approach to group key management. In *IEEE Conference on Distributed Computing Systems*, May 1998

[43] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to groups. In *ACM Symposium on Computer and communication Security*, March 1996

[44] I. Stoica, R.Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM 2001*, August 2001.

[45] Wi-Fi Alliance. Wi-Fi Protected Access (WPA). http://www.wi-fi.org.

[46] J.Yonan. OpenVPN. http://openvpn.net/, Mar. 2007.

[47] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, pp.24-30 , Nov. 1999.

[48]L. Zhu, K. Jaganathan, and K. Lauter. Elliptic curve cryptography. Support for public key cryptography for initial authentication in Kerberos. *RFC 5349*. Sep. 2008

Appendix A: Background on Cryptographic Primitives

This section presents terminology and notations that are used throughout this thesis. Cryptographic primitives like, hash functions, public-key certificates and elliptic curve cryptography are used in our solution presented in Chapters 4 and 5. The discrete logarithm problem is briefly explained to show why elliptic curve cryptography is secure. This section is based on definitions from [30].

A cryptographic hash function is informally referred to as one-way hash function or hash function. Hash functions can be used to preserve data integrity, which is of importance in emergency response systems. When violated message integrity is undetected, the network might become unstable or perform sub-optimal, because messages are interpreted incorrectly.

A hash function is a computationally efficient function, which makes it a suitable mechanism for emergency response systems that require reliable and fast communication. The function maps binary strings of arbitrary length to binary strings of a fixed length. These are called *hash-values* [30]. Hash functions are commonly used for message authentication codes to preserve data integrity. An example of a secure hash functions is SHA-1 [32].

Data integrity is preserved when the hash-value of an input is computed at time t_1, and at a subsequent point t_2 the hash-value of the same input is calculated again. If the hash-values match, the input has not changed and its integrity is preserved. A hash function should have the following properties:

- At each point in time it holds that $h(x)=h(x)$, the function is deterministic
- It should be computationally infeasible to find an x and y such that $h(x)=h(y)$, the function is collision resistant
- Given a hash-value y it is computationally infeasible to find the input, or pre-image, x such that $h(x)=y$, the function is pre-image resistant

A keyed hash function is used s message authentication code (MAC). A MAC algorithm consists of a hash function, secret key and an initialization vector. The sender of message m runs it through the MAC algorithm and produces a MAC data tag. This

tag is attached to the message and sent to the destination node. The receiver must invoke the hash function using the same key and initialization vector, to produce the same result and verify the message integrity. A MAC differs from a digital signature in the sense that the latter is unique for each node, while the former can be produced by any node having the MAC algorithm and corresponding key, because the key is symmetric. A MAC will produce a different data tag even when the message is the same, because the initialization vector is different each time the algorithm is invoked. This makes a MAC more resiliant against, for example, replay attacks. An example of a message authentication code is HMAC [10][11].

Public key encryption is a cryptographic technique that uses a key pair consisting of a public key (*PK*) and a private, or signing, key (*SK*). N_i has keys PK_i and SK_i and signs message *m* using its private key,

$$\{m\}SK_i$$

Nj can verify if the message came from N_i using the public key from N_i. This is denoted

$$\{\{m\}SK_i\}PK_i = m$$

When N_j wants to encrypt a message that only N_i can read, it uses PK_i to decrypt message *m*.

$$\{m\}PK_i$$

N_i uses its private key to decrypt the message

$$\{\{m\}PK_i\}SK_i = m$$

One way to store public keys are in a public-key certificate (PKC). A PKC consists of a *data part* and a *signature part*. The data part contains plain-text information related to the node that is owner of the certificate and must at least contain the public key and an identifier of the owner, which is usually a text string. The signature part binds the subject identity to the public key in the certificate. The signature part consists of the digital signature from a trusted third party (TTP).

A PKC makes the public key from a node available such that others can validate and verify the authenticity of the node. A node can prove it is authentic by getting its PKC signed by the TTP. To verify the authenticity of other nodes, a node must have the verification function of the signature from the TTP, which is usually the public key of the TTP. When N_i acquires the public key of N_j, it verifies the signature from the TTP its public key. When the verification is successful, N_i can accept the public key from N_j.

Symmetric encryption is a cryptographic technique that uses a single key for encryption and decryption. Symmetric encryption is denoted as

$$\{m\}K_{ij}$$

Decrypting the encrypted message results in the original plain-text message and is denoted

$$\{\{m\}K_{ij}\}K_{ji} = m,$$

where K is the symmetric key and m the message containing confidential data. A commonly used form of symmetric encryption is block cipher encryption. AES-128 is a standardized algorithm used in many standard protocols, for example 802.11i.

NONCE stands for number used once, and it is a number generated randomly. NONCEs are used to ensure that messages cannot be replayed. Nodes need to keep a list linking a NONCE to a message. When an adversary replays a message, a look up is performed to check if the NONCE is already used, and if so to what message it belonged. If there is a match, the node should drop the message considering it part of a replay attack.

Elliptic Curve Cryptography (ECC) is a public-key cryptography approach based on elliptic curves over finite fields. ECC relies on the mathematical intractability that it is infeasible to find the discrete logarithm of a random elliptic curve element,

while the base point is publicly known. Many traditional discrete logarithm protocols have been adapted to elliptic curves.

Discrete logarithm problem: In many cryptographic techniques the security depends on the intractability of the discrete logarithm problem. Discrete logarithms are the inverse of discrete exponentiations. The latter is a relative simple computation, while the former is difficult and no efficient algorithms are known or implemented. This asymmetry is of good use in cryptography, because protocols rely on fast computations that are infeasible to inverse.

The definition of a discrete logarithm requires

- G to be a cyclic group of order n,
- α to be a generator of this group and
- $\beta \in G$.

The discrete logarithm of β to the base α, is the unique integer x, where $0 \leq x \leq n-1$, such that $\beta=\alpha^x$. The definition of the DLP is as follows:

Given a prime p, a generator α of Z_p, and an element $\beta \in Z_p$*, find the integer x, $0 \leq x \leq p-2$, such that $\alpha^x = \beta$.*

For elliptic curves the discrete logarithm problem is a bit different, the elliptic curve discrete logarithm problem (ECDLP) is as follows:

Given E is an elliptic curve over a finite field F, with points P and Q lie on E and $Q \in \langle P \rangle$, find integer k, such that $Pk = Q$.

Elliptic curve Diffie-Hellman: Elliptic curve Diffie-Hellman (ECDH) is based on the Diffie-Hellman (DH) protocol [16], which allows two entities to establish a symmetric key over a non-secure channel. The only precondition is that both entities share the same domain parameters. ECDH is more efficient, has a stronger security per bit and is less vulnerable to the small subgroup attack [12] than traditional DH, which makes ECDH attractive for resource constrained mobile nodes [29].

An ECDH cryptosystem requires domain parameters, (*q*, *FR*, *a*, *b*, *G*, *n*, *h*), to be built. An elliptic curve takes it values from the order of finite field *q*, and *FR* is the field representation. The elliptic curve has coefficients *a* and *b*, base point *G*, and *nh* (where *n* is a large prime) as the number of rational points of the elliptic curve.

Consider two entities, N_i and N_j, wanting to establish a shared secret key using ECDH. Both entities have a public/private key pair, PK_i/SK_i and PK_j/SK_j. Both parties exchange their public key and compute $(x,y) = hPK_jSK_i = hPK_iSK_j$. If $(x,y) = (0,0)$ the protocol fails, else the session key is equal to $KDF(x)$, which is the key derivation function, that invokes a hash function multiple times to prevent that bits of *x* can be predicted with non-negligible advantage [28].

Appendix B: Glossary

1-affects-*n* phenomenon: When a single node joins the network or gets compromised and all *n* nodes have to perform an costly operation.

Back office: The home department of an emergency service

Back office node (BON): A node with a server like architecture that is placed in the back office

Broadcast communication: The form of communication where the message of one node is readable by multiple other nodes

Certificate authority: An entity that issues digital certificates

Client device: A laptop or PDA that belongs to an emergency responder with the possibility to communicate with mobile nodes

Communication infrastructure: The backbone of a communication systems that enables communication services to operate

Emergency responder: Policeman, fireman or paramedic

Emergency response network: A network only accessible by emergency responders. It is established and used during incidents

Emergency response service: Police, fire department or medical service

Group key: A key that is generated by the group controller and local controller, and is used by mobile nodes to perform broadcast communication

Incident: An event that requires intervention from an emergency service

Incident area: The area that is only accessible to emergency responders during an incident

Man-in-the-middle-attack: An attack whereby the attacker places itself between to entities A and B. Entity A sees the attacker as entity B and entity B sees the attacker as entity A.

Mesh key: A key that is generated by a mobile node and is used by mobile nodes to perform broadcast communication

Mobile node: A node that is mounted in emergency vehicles and establishes a network at the incident site when multiple mobile nodes are present

Node: Hardware with the capability to communicate with other nodes

Pairwise key: A key that is shared by two nodes to perform unicast communication.

Public safety party: See emergency response service

Trusted third party (TTP): An entity that is trusted by multiple communicating parties and facilitates services like authentication.

Unicast communication: The form of communication where the message of one node is readable by one other node

Virtual Private Network (VPN): A computer network that uses tunnelling protocols over a public network to create a secure network that can be remotely accessed

Appendix C: List of Abbreviations and Notations

AES: Advanced Encryption Standard

ACA: Active communication attacker

ANL: Authorized node-list

BON: Back office node

CA: Certificate authority

CRC: Cyclic redundancy check

DHIES: Diffie-Hellman Integrated Encryption Scheme

DLP: Discrete logarithm problem

DoS: Denial-of-Service

***DS_i*:** Device secret from N_i

DSA: Digital Signature Algorithm

EA: Exhaustive attacker

ECC: Elliptic curve cryptography

ECDH: Elliptic curve Diffie-Hellman

FLAME: Forwarding layer for meshing

***GC*:** Group controller

***GK_i*:** Group key generated by *GC* and N_i

***GM*:** Group member

IA: Inside attacker

***$K_{i,j}$*:** Pairwise key between N_i and N_j

***LC*:** Local controller

LAN: Local area network

MAC: M essage authentication code

MitM attack: Man-in-the-middle attack

***MK_i*:** Mesh key generated by N_i

MN: Mobile (FIGO) node

MANET: Mobile ad hoc network

NONCE: Number used once

***NS*:** Network secret

OTAR: Over the air re-keying

PCA: Passive communication attacker

***PK_i*:** Public key from N_i

PKI: Public key infrastructure

***SK_i*:** Secret or private key from N_i

SSID: Service set identifier

TTP: Trusted third party

VPN: Virutal Private Network

WMN: Wireless mesh network