NTNU
Norwegian University of
Science and Technology

Security in the MIDAS Middleware

Thomas Pronstad
Vegar Westerlund

Master of Science in Computer Science
Submission date: June 2008
Supervisor: Torbjørn Skramstad, IDI
Co-supervisor: Lillian Røstad, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

MIDAS has developed an architecture and middleware platform for solving some of the technical difficulties when developing mobile applications. Issues like network topology, low bandwidth, unreliable connections and distributed data storage are addressed by the middleware. However security needs are not addressed and no security functionality is provided.

The objective here is to:
1) Identify security needs in the MIDAS middleware
2) Evaluate possible solutions.

Assignment given: 15. January 2008
Supervisor: Torbjørn Skramstad, IDI

Abstract

Security in Mobile ad-hoc networks (MANETs) is difficult because of its operating environment and its lack of a central control unit, making classical security measures inapplicable. MIDAS is a project funded by the European Commission which creates a "Middleware platform for developing and deploying advanced mobile services". It is important for MIDAS to find a middle ground where it provides reasonable security, while using little extra processing power and battery and remains easy to use. In this thesis we identify the vulnerabilities and security measures needed to secure MIDAS, while preserving usability. We approach this problem by analysing the MIDAS design and find similarities to other known systems. From the analysis we identify threats and ethical issues, and suggest security mechanisms that solve MIDAS specific problems. The resulting security mechanisms are described in detail and tied together to create four main configurations with increasing levels of security. The configurations can then be used by MIDAS developers to implement security in a consistent way. The results are specific to MIDAS, but issues, requirements and security building blocks can be used by other projects for applicable MANET problems.

Keywords: Mobile ad-hoc network (MANET), Ethical issues, Web-of-Trust, Full distribution, No Central Authority, MIDAS

Preface

This thesis is the concluding part of our Master of Science degree. The assignment is given by the *Safety and Security in IT-systems department*, belonging to the *Department of Computer Science and Information Technology* [1] in cooperation with *SINTEF* [2]. We both have an interest for security in computer science and have focused the concluding part of our studies on security topics, which is why we chose to write this thesis. The thesis has a timeframe of 20 weeks from start to delivery and must produce a concrete result. In our case the result is manifested in the form of requirements and configurations that can secure MIDAS.

We have received weekly guidance from Leendert W. M. Wienhofen, Inger Anne Tøndel and overall guidance from Lillian Røstad. We would like to thank them for their suggestions, feedback and helpfullness. We would further like to thank Ulrik Johansen for his MIDAS fact checking and members of the MIDAS project for various feedback to questions.

June 13, 2008

Thomas Pronstad

Vegar Westerlund

[1] Web page: http://www.idi.ntnu.no/

[2] SINTEF is an independent research organisation with a *Information and Communication Technology division* with a *department of Software Engineering, Safety and Security*. Web page: http://www.sintef.no

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The aim of the IST-MIDAS project is to develop an application middleware platform to "simplify and speed up the task of developing and deploying mobile applications and services"[1]. The project is funded by the European Commission. It started January 2006 and will end September 2008.

The project has a pure research agenda which later will lead to a commercial product based on the results.

The early MIDAS project mandate states that security is out of scope and should be solved in a parallel or consecutive project. Some work has been done investigating how to protect the routing protocol and network traffic using encrypted messages. This will however not suffice when dynamic extension of the network is needed. If the system is to be accepted and used more work should be done.

The MIDAS middleware aims to give application developers a stable Application Programming Interface (API) to exchange messages and retrieve information from an ad-hoc network while making the specifics of the network technology and topology transparent to the developer.

Since ad-hoc networks usually are wireless, as is the case with MIDAS, eavesdropping and packet injection is easier than with wired networks. This makes security more difficult, especially if sensitive information is transmitted. In this report we identify the most exposed security issues in the MIDAS middleware, look at ethical issues regarding information available in the network and look at ways to communicate securely on an ad-hoc basis.

1.2 Problem statement

> *MIDAS has developed an architecture and middleware platform for solving some of the technical difficulties when developing mobile applications. Issues like network topology, low bandwidth, unreliable connections and distributed data*

[1] http://www.ist-midas.org/

storage are addressed by the middleware. However security needs are not addressed and no security functionality is provided.

The objective here is to 1) Identify security needs in the MIDAS middleware, 2) Evaluate possible solutions.

1.3 Research goals

The main goal of this thesis is to identify the security issues in MIDAS and to assess how the security level can be raised to an appropriate level. It is important to identify changes that affect the design or architecture as these are hard to implement on a running system. The earlier these changes are adapted the cheaper they are to fix [23]. We will investigate what vulnerabilities exist in MIDAS, and define a reasonable border of responsibility between the middleware and applications running on top of it.

We will achieve these goals by defining security requirements for the system, research similar systems and solutions, propose a model for how security can be implemented and identify architectural changes, if any. The model will be validated together with the MIDAS research group. Which requirements have been met and how, which trade-offs where made and what new problems emerged will be discussed.

1.3.1 Ethical issues and incentives for security

Ethical issues in this thesis are issues created by the MIDAS software in its interaction with people. We have devoted the first part of Chapter 4 to ethical issues where MIDAS specific issues are identified. We will use a general framework to find the issues and put them in context. The identified issues should be a background and a motivational factor for security.

The ethical issues should also discuss what makes people share their personal information. This is a supporting section to the basic ethical issues which emphasises the responsibility the domain responsible[2] has when making a MIDAS service. We would also like to point out the factors that can make it easier to achieve success with MIDAS driven services.

The second part of the incentives for security is the security analysis and the threat model. The security analysis should reveal specific weak points of the MIDAS architecture. The threat model should give a clear picture of the possible types of threats to MIDAS applications. Together they will be the basis for the security requirements.

1.3.2 Security solution

The goals for the security solution part of the thesis is to suggest security mechanisms that together can solve the security requirements. The solution shall raise the level of

[2]See Section 3.4 for explanation of domain responsible

security by applying security mechanisms to specific parts while maintaining a view of the entire system. The described mechanisms in the security solution can consist of mechanisms we suggest that are new, or mechanisms from other systems.

1.3.3 Level of security supported by middleware

Achieving military grade security in the MIDAS middleware is in itself not desirable, because MIDAS is to be used commercially and usability has to be balanced with the level of security; it needs to be "secure enough". Since MIDAS is a middleware, platform security might not be an issue for the service itself. "Secure enough" might be no security at all if all the information generated in the network is considered public.

Applications and protocols can be made secure on top of insecure infrastructure. Examples include S/MIME [16] for securing email and kerberos [26] for authentication over unencrypted links. This means that security mechanisms provided by the middleware for the applications to use do not have to cover all thinkable demands. If the specific service domain requires stronger security than the middleware can provide, it has to be achieved in the applications. Making this trade-off is important for the middleware to be usable. If for instance all traffic was encrypted and only trusted nodes were allowed to participate in the network the middleware would be rendered useless for spectators of a sporting event.

1.4 Research method

We have chosen to perform a case study of the MIDAS middleware. The case study begins with an examination of system documentation to obtain a deeper understanding of how the middleware works. When the deeper understanding of the system is in place we use the resulting mental model to generalise about the system and find similarities with other systems.

Since this is a security analysis we use prior knowledge of security patterns and vulnerabilities to identify potential weaknesses in MIDAS. These weaknesses in combination with threats to the system give rise to security requirements, which then are incorporated into a solution aiming to mitigate risks and mend weaknesses.

The case study evolves through this thesis and its progress is reflected chronologically through the chapters. The last part of the thesis evaluates and discusses the results of the case study, and is thus technically not a part of the case study.

1.5 Scope

This thesis investigates and suggests appropriate security measures for MIDAS. We give a complete solution on how we would incorporate security in the MIDAS architecture. We will not discuss mathematical strengths of cryptographic algorithms and not give exact key lengths, protocols or algorithms. Instead we are flexible, leaving it to the MIDAS developers to select algorithms and such, we suggest one technology,

but show how it can be replaced. We do this in part because of the uncertainty as to when MIDAS will be in production and how well it will manage performance issues. Advice on key lengths, algorithms and technologies can easily be found later, and it is more appropriate to choose these functional third party solutions when security is introduced, rather than now, because experience shows that as computing power increases and new attacks surfaces, algorithms and key lengths must be reconsidered.

We do not interfere or question issues that the MIDAS documentation states it will solve. These issues include: Common notion of time, Database consistency in network partitions, Database synchronisation, Reliability and Availability, Network setup, Node addressing and routing. These issues are mentioned in various parts of the thesis in relation to security, when security measures might influence them.

1.6 Report outline

- Chapter 1 - Introduction
- Chapter 2 - Peer-to-peer networks
- Chapter 3 - The MIDAS middleware
- Chapter 4 - Incentives for security
- Chapter 5 - MIDAS security requirements
- Chapter 6 - Security building blocks
- Chapter 7 - Different configurations
- Chapter 8 - Evaluation and Discussion
- Chapter 9 - Conclusion and Future work

Chapter 2, 3 and 4 represent the state of the art and related work. Section 4.3, Security analysis of the architecture is the transition from state of the art to own contribution. Chapter 5 defines the security requirements in the MIDAS middleware based on our analysis of the architecture, concerns identified by the MIDAS project and our ethical discussion. Chapter 6 presents our suggestions of security building blocks that can be used to implement the identified security requirements and Chapter 7 presents different configurations of these building blocks to achieve different levels of security.

Chapter 8 evaluates our work and discusses the solution with regards to feasibility of integration, simple performance characteristics, benefits, pitfalls and the value of our research. The last chapter concludes the thesis and sums up future work that has been pointed out through the thesis.

Chapter 2

Peer-to-peer networks

This chapter introduces properties of different P2P networks, look at typical security problems and refer to related work on security.

2.1 Peer-to-peer networks

P2P networks are non-hierarchical and characterised by peers which cooperate to achieve a common service. Since cooperation is essential, every node has to have some interest in a working service.

In a P2P network there is usually no notion of servers, the nodes act as both client and server at the same time. One beneficial property of these networks is that they can be made to scale linearly with the number of users in the network as each new participant also contribute to the overall resources in the network. This includes storage capacity, bandwidth and processing power. Another advantage with P2P networks is the robustness of random failures. Nodes come in and out of reach all the time, so the networks are configured to handle these types of situations. Typical uses are file sharing networks[1] (for efficient searching and distribution of content), anonymising networks[2] and media streaming [59].

2.2 MANET

A MANET is a self configuring network of nodes connected by wireless links and a special form of P2P. In itself it only provides the ability for peers to communicate, so to be of real value it must have a purpose and provide a service/solve a problem for the participating nodes. The One Laptop per Child [3] project is an excellent example of MANETs where the laptops form a network among themselves where no infrastructure is available.

[1] Gnutella http://en.wikipedia.org/wiki/Gnutella
Napster http://en.wikipedia.org/wiki/Napster
[2] Tor network http://www.torproject.org/
[3] One laptop per child network cooperation. http://laptop.org/

As Personal Digital Assistants (PDAs) and smartphones become more and more common, MANETs become increasingly popular and their range of applications increase. MANETs have the advantage of providing network services without an existing infrastructure, and can be setup on very short notice. This enables mobile devices to create and join networks at will, supporting services anywhere and anytime [55].

Because MANETS do not rely on existing infrastructure and has minimal configuration needs it is ideal for emergency situations and military operations. This was pointed out by Yi and Kravets [60].

2.3 Context-aware pervasive systems

Context aware systems sense their environment and are able to react to changes in it [46]. In this thesis we use context as a common denominator of personal states. Context usually includes location, nearby people and resources, but mood, social setting and availability also apply. Context is dynamic and it changes in line with the environment.

Context aware computing is a form of pervasive (or ubiquitous) computing [46]. In pervasive systems, devices are incorporated in everyday objects and interconnection between them make new applications possible. Humans can interact with these devices without even knowing that they are there.

Generating new services in context-aware pervasive systems have been heavily researched in later years. The possibilities for a new area of computer usage are enormous, but really useful applications have yet to appear. The field needs a groundbreaking application to reach large scale use.

As we will see later, there are also some privacy issues involved with connected pervasive systems that need to be handled.

2.4 Storage in P2P networks

A common area of application for P2P networks is data storage and distribution, and several schemes have seen widespread use. The fault tolerant properties of these networks ensure both availability and integrity by replicating the data [10].

One widely researched scheme is Distributed Hash Tables (DHT) [42] [50] which tries to maximise availability while minimising network traffic and node traversing in searches. DHTs generates a structured topology in the network by arranging nodes in a ring and distributing keys (and data) along this ring. Smart algorithms ensure that looking up keys can be done fast and reliable.

Other schemes (as used in MIDAS) have a distributed database where replica nodes are scattered in the network sharing the load. Depending on how fast the network topology changes (nodes entering and leaving) the number of replicas has to be adjusted to balance network traffic with availability.

2.5 Security in mobile P2P networks

Mobile peer-to-peer networks are characterised by having no central control unit making classical network security measures inapplicable [44, p. 33] and they use wireless links which is easy to both eavesdrop and inject packages into [5]. Also, devices are small and are easily lost or stolen [22]. This implies that ensuring security for sensitive information in these networks is difficult. The fact that the devices have limited processing capabilities and battery life means that strong cryptographic schemes might have to be avoided.

The threats that affect MANETs have been categorised Hubaux et al. [22] as:

- Attacks on basic mechanisms
- Attacks on the security mechanisms.

Basic mechanisms include communication links (wireless), captured and compromised devices, node behavior and routing mechanisms [29]. Threats against stored data include deletion, data modification, denying data existence and Denial of Service (DoS) by introducing too much data into the system.

Attacks against the routing protocol in MANETS are described by Sarkar et al. [44, pp. 25-27]:

Rushing attack attacks the node discovery phase of the routing protocol and tries to become a central hub in the network.

Black hole attack drops packages to disrupt service (often combined with the rushing attack described above)

Neighbour attack the attack involve failing to update the route hop part of messages when forwarding packets, making the next node believe it is in direct contact with the previous node.

Jellyfish attack delays packages instead of dropping them as in the black hole attack creating unnecessary long message delays.

Security mechanisms are system dependent, but often include cryptographic systems, access control schemes, signatures and the like.

Chapter 3

The MIDAS middleware

This chapter explains the MIDAS architecture and its components. Proof of Concept (PoC) scenarios and key challenges are introduced as well as related security work done in the project.

3.1 The MIDAS project

The IST-MIDAS project is funded by the European commission and belongs to the program "Mobile and Wireless System and Platforms beyond 3G". The project shall implement a platform that:

> *Simplify and speed up the task of developing and deploying mobile applications and services [31].*

The MIDAS middleware consists of an overall architecture, and middleware building blocks providing solutions to technical issues that must be addressed in developing mobile solutions (e.g. common connectivity issues, data retrieval methods and context aware optimisations). The middleware is used by application developers that produce a collection of applications (with instances running on multiple nodes), which together implement a mobile service[1]. The mobile service is customised for a particular event (of limited duration) and is fundamentally based on supporting mobile information sharing.

3.2 MIDAS proof of concept scenarios

The MIDAS project presents two proof of concept scenarios as an illustration of its usefulness. The two scenarios are selected to reflect two completely different

[1]There is some inconsistency within the research community about the terms "service" and "application". In the Telecoms community a service is typically something achieved using applications, but in the Software Engineering community - using the Service Oriented Architecture (SOA) approach - applications are realized using services. Note that, in MIDAS, we adopt the Telecoms style, basing this on the terminology dictionary [34] published by the Open Mobile Alliance (OMA) [35].

extremities, which give rise to different sets of requirements and is used to validate the solution. The scenarios are described in detail in MIDAS Deliverable 1.2 [54, pp. 15-19].

3.2.1 Emergency scenario

The scenario describe a fire on the Paris Metro. The fire is detected by the train driver which issues the fire emergency procedure from his MIDAS enabled device. Location and other attributes of interest are automatically sent to the emergency headquarter. All MIDAS enabled devices nearby receive information about the emergency based on the context the device is in. Emergency personnel arriving at the scene will be requested to join the emergency network and after responding to the request the device will be configured with the necessary applications and start receiving messages. Because the wireless infrastructure is limited under ground, MIDAS devices quickly extend the reach of existing network coverage.

Key features in this scenario is:

- More reliable communication in places where infrastructure is non-existent or knocked out.
- Easy setup and configuration of user nodes.
- Context based addressing makes organizing easier.
- Logging for after the fact evaluation of the emergency.

3.2.2 Sports scenario

The sport event chosen is Tour de France. This event has some unique challenges due to the divergency and number of users and the location of the event. The location changes every day and ranges from populated city areas to mountain tops. Cyclists should be tracked in real time with bio information and geological position and speed. The cyclists current position in the race combined with overall ranking should also be combined to see how the race will affect the results.

The users of the system have different requirements based on their role (e.g. *VIP*, *team manager*, *journalist*, *spectator*, *police officer* or other). A spectator will for instance be interested in being notified 5 minutes in advance of his favourite cyclist passing him. This notification request will have to take into account the position of the spectator (approximately) and the position and speed of the cyclist.

Key features in this scenario is:

- Transparent network communication when a large number of users and multiple networks are present.
- Instant infrastructure setup where none exists.
- Context information and context notification capabilities.

3.3 MIDAS middleware architecture

This section presents the MIDAS architecture. MIDAS follows a modular architecture where well defined interfaces between the components makes it possible to reimplement each module isolated from the others. Figure 3.1 shows the architecture in a component interaction diagram. The description is based upon MIDAS Deliverable 2.2 Part 2 [36].

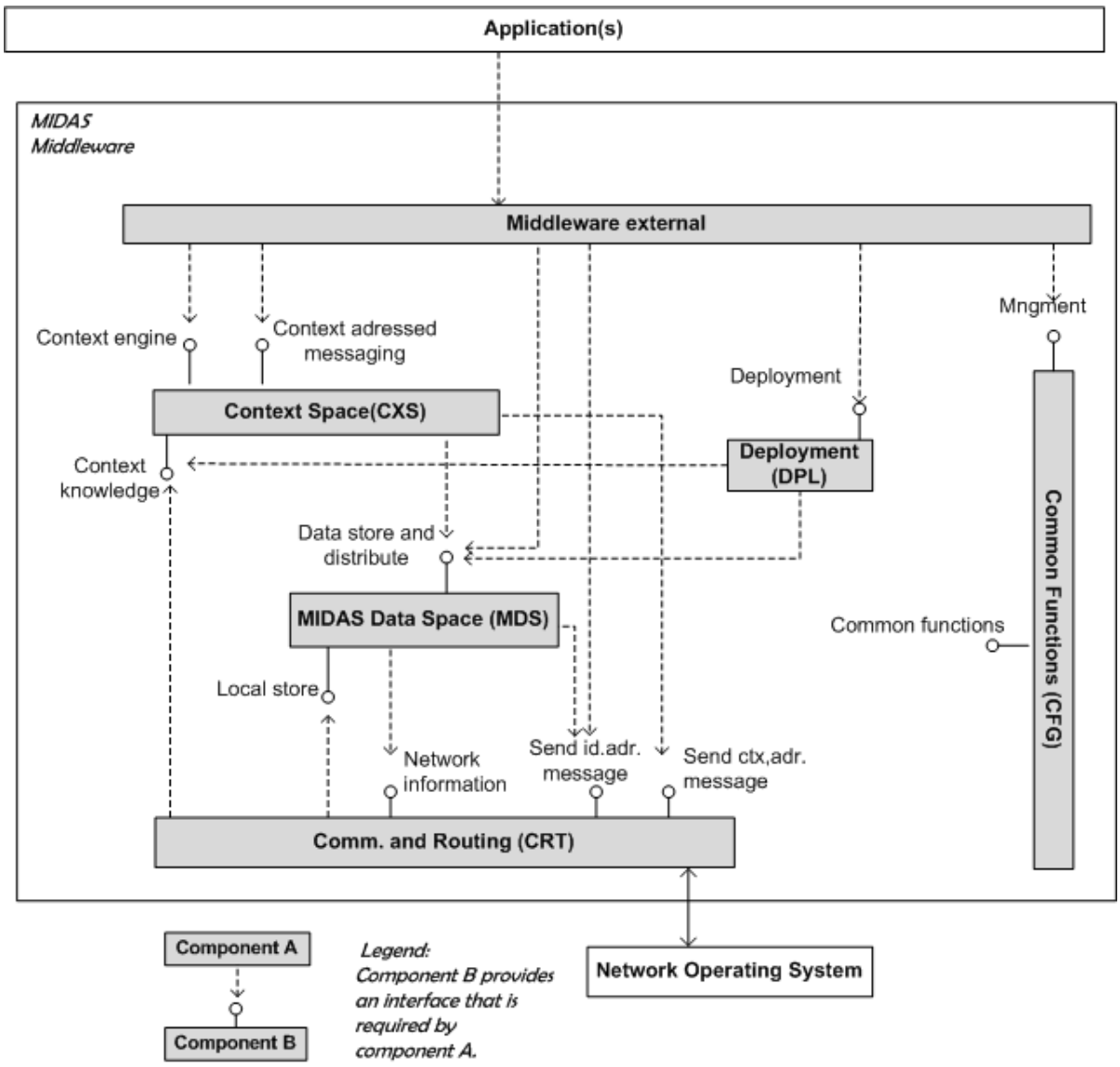

Figure 3.1: MIDAS architectural overview

The MIDAS middleware is divided into 5 components.

- MIDAS Data Space (MDS) Component
- Communication and routing (CRT) Component
- Context Space (CXS) Component
- Deployment (DPL) Component
- Common Functions (CFG) Component.

3.3.1 MIDAS Data Space

The MIDAS Data Space component is characterized by the following features:

- MDS stores and retrieves data from potentially several nodes
- MDS provides a notification service for applications which alerts them when data is modified.
- MDS manages the replication and distribution of data.

The MDS consists of a top layer which does Query Analysis, a Data Allocator which distributes data across an unknown amount of nodes each containing a local database, and service components which enable queries, mask the distribution for the developer and synchronise data. To aid synchronisation and the provided consistency, deletion of data is not supported. Instead the status flag in the metadata is updated to reflect the delete operation.

3.3.2 Communication and routing

The Communication and Routing component is characterized by the following features:

- CRT makes the transport medium transparent for the application developer.
- CRT will store transmitted messages for nodes not available in the network for the time specified in the time to live (TTL).
- CRT will use a distributed routing table scheme to route packages.
- Context based routing (CBR) will use both CXS functionality and CRT functionality to route context addressed packages. It is defined as a part of the CRT component.

The CRT component consists of a *Message Dispatcher*, *Packet Forwarder*, *Network Manager*, *Context Based Router*, *Node Identifier Solver* and *CRT Manager*. From a security standpoint the most interesting subcomponents are the *Packet Forwarder* which determine the best network medium to use and *The Node Identifier Solver* which accesses routing tables and requests updates from *Central Node* if information is unavailable. The other components perform tasks like storing messages and interfacing with network interfaces.

3.3.3 Context Space

The CXS Component is characterized by the following features:

- Context knowledge base (CKB) stores context information and contains the static domain model and dynamic context data. This component provides reasoning. The model and context data are stored in MDS.
- Context engine (CTX) provides context triggers and context queries which aggregate and derive new knowledge.
- CAM is the facade which uses CKB and CTX to provide the Context Addressable Messaging service.

The CXS component has the three sub components CKB, CTX and CAM. Security issues with CTX includes privacy and integrity of context information. Addresses generated by CAM are based on context information distributed to all nodes by the routing protocol. This results in a trade-off between addressing and revealing information.

3.3.4 Deployment Module

The Deployment Module Component is responsible for distribution, configuration and installation of applications either with user interaction or when context conditions are met.

The context conditions enable DPL to install predefined packages seamlessly when a user enters a new context.

- DPL will use MDS to store software bundles.
- DPL will register triggers in CTX for context dependencies.
- DPL will download, install and manage all MIDAS applications.

3.3.5 Common Functions

CFG is the component containing all the functionality which doesn't fit into the other components. CFG is intended to have functions that manage time, battery and power, error handling, local logging and identify storage resources on the device.

Common notion of time related to the lack of a global clock poses a possible security issue in MIDAS, but according to requirements from Deliverable 1.2 Part IV [52], a solution of relative time between nodes is being discussed. We are aware of the problem of global time, though in this thesis we will disregard it, since no solution for time management is decided upon at the time of writing.

3.4 Service domains

Services implemented on the MIDAS middleware are always defined for a given service domain. The *sports* and *emergency* scenarios from Section 3.2 are examples of different service domains with very different needs.

For each service domain there exists a domain ontology describing entities in the system and how they interact. All services within the same network share this common ontology.

Specification of a domain ontology model are initiated by the *domain responsible* which is usually an organisation who want to develop mobile services within its domain. Great care should be taken when specifying such a model since it will be shared by all services running in the network.

3.5 Types and roles of nodes in MIDAS

The MIDAS network are built up of different types of nodes, which again can be assigned roles. The different types are described in MIDAS Deliverable 1.2 Part I [54, pp. 35-38]. The node types and roles interesting to our evaluation are described here.

MIDAS node is the definition of a node with the core MIDAS components installed, making it able to take part in the network. A *user node* is the most common type of MIDAS node, which has different applications installed based on the given service domain and current context. One or more *central node(s)* are required in all MIDAS networks (or partitions) to keep track of which other MIDAS nodes are present. If no central node are present this role will be assigned by the CRT component. The *deployment administrator node* supply user nodes with packages (configuration settings and software applications) when users request to take part in a service.

3.6 Key challenges in MIDAS

MIDAS solves the problem of network addressing complexity in MANETs and at the same time support given service domains with ontology rules for reasoning and a global shared data store for information sharing. Key challenges according to the projects own definition are [31]:

- Large number of users
- Network must be setup on short notice in ad-hoc fashion

- Pre-existing infrastructure limited
- Dynamic network topology

3.7 Proposed security approaches in MIDAS

Some work has been done previously with assessing the security needs in MIDAS. Our solution is based upon this work, trying to eliminate some of the limitations found. The proposed solution is intended for emergency scenario, but might work in the sports scenario as well.

3.7.1 Proposed security approach for sport event scenario

We do not know of any previously proposed security approaches for sport event services on top of the MIDAS middleware. Our proposed solution also cover this scenario.

3.7.2 Proposed security approach for emergency scenario

Puzar et al. proposes a security solution for an emergency service [38] running on top of MIDAS. They combine a bottom-up and top-down approach to secure network traffic including the routing protocol, and ensure application security by message signing.

Puzar makes the following assumptions:

- Members of the rescue personnel (users) can be trusted.
- Devices are preconfigured before coming to the scene; this includes installation of the necessary software, certificates, etc.
- Users carry tokens with personal certificates containing their credentials (name, rank / role within the organisation, etc.) and use them to authenticate themselves towards the service.
- The user-to-device authentication process is assumed to be secure enough
- Software developed on top of the MIDAS middleware is considered clean (that is, it is not vulnerable, there are no viruses, trojans, etc.)

Given these assumptions Puzar et al. propose a solution which is reasonably secure. Their solution uses X.509 certificates that are signed and installed prior to the emergency situation. These certificates are also used to prove rank / roles within organisation and can be validated by all nodes considered a part of the network (certificates are ultimately signed by the same Certificate authority (CA)). Puzar utilizes this to establish a symmetric group key using the SKiMPy key protocol [37]. This key is used to encrypt traffic in the network and nodes with this key make up

a core network that routes packages. Untrusted nodes are excluded, but temporary certificates can be issued to nodes manually. This will make the node a *passive* member of the core routing network, able to participate but not able to change the routing tables or act as a storage node. Messages are also signed or encrypted using the private or public key of the X.509 certificates to provide application security.

The assumptions of pre-installed certificates and trusted users will not hold in all situations where MIDAS could be used, as the design is limiting. Spectators in the sports scenario will not have a trusted certificate and will therefore not be able to take part in the core routing network loosing the benefits of a MANET. The hierarchical trust model (X.509 can only have one root-certificate) will also be a problem where several organisations cooperate to generate services in the network. In Section 6.1 we will look at how some of these limitations can be removed by simple changes to the design and what the consequences of these changes are.

Chapter 4

Incentives for security

There are several incentives for security in the MIDAS network:

- Sensitive information can be available and must be handled in an ethically secure way.
- Laws and regulations demand that personal information is kept private.
- Software that make users confident that it is secure is more likely to be accepted and reach widespread usage (which is a definite goal in MIDAS)
- The systems normal operation and behavior is dependent upon the system state which should be protected from malicious modification.

The first three incentives will be discussed in the first part of this chapter. Ethical issues that arise in human computer interaction are also discussed to determine what people find acceptable or not. This is a broad topic with several different viewpoints, but we limit our discussion to issues closely related to how MIDAS gather information and how this affects people. Acceptance of the application relies on this.

The second part will make a analysis of the architecture to identify weaknesses in the system, and define a threat model that describes possible attackers in the MIDAS network. The goal here is to find what effects the normal operation of the system.

This chapter will be the basis for the security requirements in the next chapter.

4.1 Ethical issues in pervasive systems

What identifies ethical issues is a concern about how computers affect humans. The field of context aware and pervasive systems is relatively new. Ever since the first types of systems, designers noticed the new kind of ethical issues resulting from new system behaviour, namely the localisation factor. Roy Want describes the press attention stirred when introducing a call routing system in the early 90's in his article [57]. The system used badges which identified persons and coupled it with their location to route incoming calls to the nearest telephone. He found that the interpretation of right-to-privacy is strongly connected with the social setting in which the new system

is being used. People accepted the system because it helped them in their work. Khalil and Connely show further in their study [25] of cell phone context information[1] that this interpretation is connected with where we are and who is seeking us. They found that people are likely to reveal information when it will be beneficial to themselves. New systems therefore need to provide a useful service to be accepted among the general public.

The two articles cited above highlight common issues in pervasive systems, namely privacy and information disclosure. These issues are regulated by two factors; legal requirements and public opinion. Milne [33] highlights the connection between the marketer [2] and consumer[3] interaction and lists some of the influences that they have. Figure 4.1 shows the influences and interactions between marketer and consumer as they fit into the framework proposed by Milne.

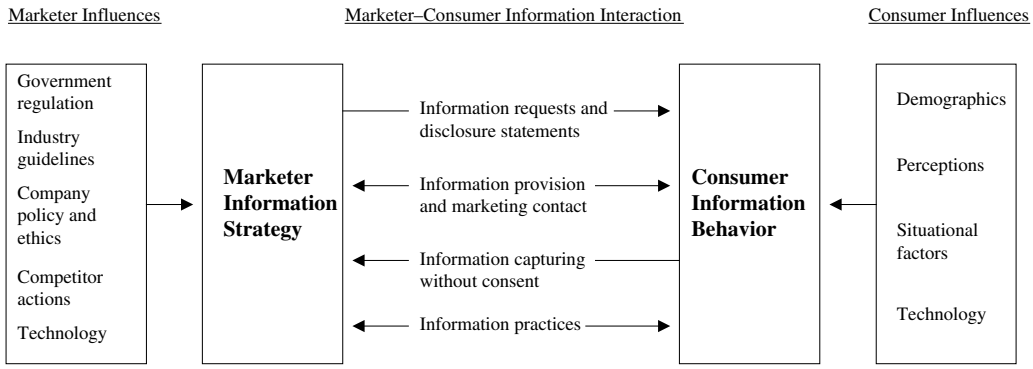

Figure 4.1: Milnes ethical issues framework [4].

For pervasive systems and MANETs in particular, influences that affect how privacy is perceived and handled are: Government regulation, Company policy and ethics, Technology, Perceptions and Situational factors. MIDAS must take this into account when designing a solution that will handle personal information.

4.1.1 Localisation

As MIDAS uses wireless devices, localisation of a user is possible. Physical position is considered private and disclosure of this information becomes an issue in MIDAS.

Localisation of users in MIDAS is threatened because of at least two factors: Signal detection and route tracing. Signal detection is a problem that is hard to protect against. The problem arises because the small differences of Radio Frequency (RF) waves in a radio transmitter can be used to identify the device much like a fingerprint [57]. Remley et al. have tried to use this fingerprint for device authentication and

[1]Cell phone context information in this case was where a people were and what they were doing
[2]Marketer is the person or company that provides a service and has ownership over the computer system. The MIDAS equivalent is the domain responsible.
[3]Consumer is the person which uses a service and contribute information to the system. The MIDAS equivalent is a user.
[4]Figure copied from Milne [33]

their test shows that in a controlled environment it is possible to positively identify a transmitter [41]. Protection against this threat will be nearly impossible to implement in MIDAS since we cannot control the RF transmitter at this level. However, the information can only be used to fingerprint a device and locate it when its signal is within reach of an attackers antenna and a connection between a user and the device cannot be made solely with this information.

Route tracing is an attack on the network protocol which can be used to see how data from one node travels through the network. By analysis of the routing table and the network traffic one node transmits, it is possible to locate the node. The routing table tells which nodes are nearby and how many intermediate nodes messages go through. For MIDAS this means that specific node identifiers can be located by analysis of the network protocol and transmission of network packets.

4.1.2 Personal and sensitive information

Personal information, whether it is medical history, social security numbers or other secrets is private and should be protected from unauthorized access. For the emergency scenario this kind of information will be useful for rescue personnel, but must be handled with care. In MIDAS there will be a domain responsible which has to decide what information should be possible to collect in the system and how this information should be treated. This will be defined in the ontology model[5].

Using the two scenarios presented in Section 3.2 as a basis, sensitive information can range from medical history to strategic sport information. What information is regarded as sensitive will be decided by the influences which was presented in the privacy framework earlier. The prominent influences will be government regulations, perceptions and situational factors. Information gathered about individuals is personal and will usually be perceived as sensitive. Regulations will state what types of information the domain responsible can gather, the level of sensitivity the data has and for which purposes it can be used. The situational factors will often determine whether it is acceptable to use information or not. As an example it is, generally speaking, OK to use information about your medical history if you are injured, but it is not OK to use that same information for a news paper article.

For a domain responsible two other factors are important, namely legal requirements and reputation. Reputation is important for companies and bad press resulting from an application weakness can be expensive. This reflects back on corporate policy and ethics. Legal requirements imposed by the government obviously have to be followed.

Norwegian law determines that personal information do not belong to the person or organisation who collected it, but to the person the information is about. This means that information regarding an individual should be deleted at this individuals request.

4.1.3 Sensitive context information

Context information in MIDAS is information about what you are doing, where you are, who you are, sensor data like temperature, capabilities you possess and other

[5]See Section 3.4 for information about domain responsible and the ontology model

things that can be useful in a domain. This type of information is related to persons and describes their state. In addition to the obvious threats to privacy by information in your context, aggregated information threats also have to be considered. MIDAS can aggregate context information which can create new and unforeseen privacy issues. This means that the domain responsible must be aware of and investigate these possibilities and create policies to regulate the information that is gathered.

4.2 User acceptance and sharing of information

The domain responsible wants to give his customers a useful service and make it popular, which is an important goal for the service if it is to be deemed successful. In order for context aware services to be accepted by people they must address a perceived need and be easy to use [12]. This leads to usability requirements on security functionality. Barkhuus [3] also identifies coolness as a factor of user acceptance. This means that from a user perspective there exists a trade-off between privacy and coolness.[6]

MIDAS represents something new and has potential for a high coolness factor. This potential exists because MIDAS can provide network services in previously disconnected areas and have applications that benefits from this. As an example we envision an application where users cooperate to create information in the form of videos, text and audio covering a sports event where conventional channels of information are unavailable.

The study presented by Cranor et. al. [28, pp. 47-55] shows that users are more likely to share information if they are anonymous. This means that some applications can increase their usage by not connecting contributed information with users. To influence this factor it would be beneficial if users were able to select who (among their personal friends) would be able to view the information they share. In another study, Acquisti shows that even Facebook users claiming to be privacy aware reveal a lot of private information, and awareness is only weakly related to the amount of information that is shared [2].

This means that if people are given the opportunity to share private information they might do so, not thinking about the consequences. The coolness factor increases what people are willing to share, making it possible to achieve the desired usage and information sharing needed for an application to be successful. The domain responsible must weigh these two factors against each other, determine what the system needs of information and create routines that preserve reasonable privacy.

4.3 Security analysis of the architecture

Security is not emphasized in the MIDAS project, however, it was stated that the architecture should be designed in a way so that security could be implemented without changing the overall structure of the system. The individual components in MIDAS export well-defined interfaces for communication and flow of control.

[6]Coolness is a perceived positive factor of novel usage potential

However, none of the interfaces have any kind of access control or validity checks. The system assumes that nodes are well behaved. From looking at the architecture overview (Figure 3.1 in Section 3.3) we see that all the major components use MDS for storing and retrieving data. The fact that MDS is both readable and writeable for every node in the network and used by all the other components makes it a logical target for attacks.

The routing protocol used in MIDAS is also vulnerable since a malicious or malfunctioning node will be able to change the state of the routing tables causing disruption in the network. Attacks here might bring down entire parts of the network resulting in Denial of Service.

The MIDAS network is used to route messages from one node to another via a set of intermediate peer nodes. These messages are transmitted in clear over wireless links and are prone to modification and information disclosure attacks. Any node capable of receiving the signal (and of course the intermediate nodes) will be able to read the message sent. If private or other sensitive information is to be sent in this network this will have to be mended.

The initial analysis of the architecture shows, three areas that stand out as vulnerable; MDS, routing, and messaging. A security in depth solution for this system must at least address these areas.

4.4 Threat model

To assess the security in MIDAS we need to identify the threats to the system. We will in this section briefly describe these threats, probable attacks and the capabilities of an adversary. Diaz et al. [13] classifies attackers in the following categories:

- Internal – external
- Passive – active
- Local – global

The *internal attacker* in MIDAS would be an insider with trust already established in the system. Insiders would use this trust to gather information available in the network in order to use it with malicious intent. Disrupting services might also be easier for an insider depending on whether the network security is based on controlling the border (only) or a security in depth solution is used. An *external attacker* would be somebody which is previously unknown in the network, but who joins the network with malicious intent. He has no established trust so he will have to gain trust or limit attacks to vectors not involving trust.

A *passive attack* would mean to eavesdrop on the network traffic gathering as much information as possible. Sifting through messages addressed to other nodes or even just registering source and destination nodes for messages could provide valuable information to an attacker. *Active attacks* go further and inject, modify or drop packets in transit in order to actively alter other nodes perception of reality. Because of this, active attacks are potentially more devastating.

The difference between a *local* and a *global* attacker is how large a portion of the system he can see and control. A *global* attacker sees the entire network and can manipulate the routing protocol or inject fake packets. Normally users of MIDAS would only 'see' one-hop neighbours and therefore have a local view. An attacker with a single device will be classified as local.

Attackers can also be classified according to how risk averse they are [47, pp. 59-72]. The more risk an attacker is willing to take the harder it will be to defend the system.

- Disgruntled employee (insider)
- Determined attacker (outsider)
- Script kiddie (outsider with little resources)

A *disgruntled employee* have intimate knowledge of the system and some level of access, making him a powerful adversary. However, his risk aversion might range from low to high, based on why he is disgruntled. He might not be ready to risk losing his job, only hurt the company as much as possible. A *determined attacker* is usually more resourceful than an insider, have more experience and is normally anonymous making him less risk averse. The *script kiddie* has the least amount of resources and is mostly hacking for fun. He does not necessarily believe or understand that what he does is wrong and therefore neglects the risk of being caught.

Based on these attacker profiles we see several possible threats that should be mitigated.

- Attacking the routing protocol
  - Flooding the network with traffic
  - Replay or inject packets
  - Dropping packets
  - Disrupting the network, denying services to other users
- Eavesdropping, gather information
- Trying to elevate permissions, gaining access to sensitive information
- Receive services not granted to user (for personal benefit or economic gain)
- Spam

Attacking the routing protocol and eavesdropping the network are general threats that apply to all MIDAS scenarios where malicious nodes might be present. Elevating permissions and receiving services are only possible where restricted services or sensitive information is available in the network, depending on the scenario in question. Sending spam for commercial or political reasons is for example applicable where spectators are involved.

4.5 Summation of different issues in MIDAS

We have seen in this chapter that there are security issues with MIDAS architecture and the inherent properties of MANETs. The following is a list of issues which sums up the different security incentives:(These will be addressed in the security requirements and through the remaining chapters)

Privacy Sensitive information must be protected, but how should sensitive information be identified and for how long must it stay private? What information is considered sensitive will vary between service domain models. Since there is no central server in MIDAS every node has to evaluate and verify this individually.

Personal information MIDAS will store and aggregate personal information for participants of the network. According to Norwegian laws[7] this information has to be deleted at the participants request. Personal information includes location data which will always be available in MIDAS.

Untrusted network topology The network is dynamic and packages will most likely be sent over untrusted channels.

Ad-hoc routing Routing tables will change, nodes will leave, routing information must be updated and distributed without being maliciously manipulated.

Lack of Access Control Who is in charge of Access Control? Who owns and control information? No central authority will be in place to handle this.

Denial of Service Denial of service can be fatal in emergency and rescue and costly in sports events.

Usability The middleware must be easy to use and allow dynamic setup.

Statistical information Statistical information becomes valuable when a large number of users connect. The information could be correlated with other data to identify individuals and used in fraud attempts.

Trust in nodes Regular nodes in the network might not be trusted, but should still be able to handle data or route messages.

Data integrity The global database is distributed on untrusted nodes, with no central control enforcing integrity and consistency.

The issues presented above are the main issues so far. They come from the design of MIDAS and from the environment MIDAS will operate in.

[7]Norwegian laws are compliant to EU laws and in regard to privacy regarded as stricter

4.6 How issues affect the thesis

The ethical issues presented in Section 4.1 give rise to ethical requirements presented in Chapter 5. Other concerns regarding the architecture and overall security issues are used to determine what functional security is needed. They are also one of the reasons security is needed in MIDAS. Even if there was no economic incentive for security, influences like government regulations and public perceptions would still demand a certain level of security.

This chapter has also shown how applications on the MIDAS middleware can benefit from a coolness factor and get users exited about the product. Ethical issues is discussed and evaluated in respect to suggested security measures in Chapter 8.

Chapter 5

MIDAS security requirements

Before making reasonable design decisions about security we need to identify requirements for the services running on the middleware. We present here a list of requirements for different parts of the system. The requirements are based on the challenges described in MIDAS Deliverable 1.2 part III [53] and issues identified in Section 4. Each requirement is given a priority ranging from high **(H)**, medium **(M)** and low **(L)**, based on our subjective opinion and evaluation of PoC scenarios. The list has been verified by members of the MIDAS project. Challenges from MIDAS Deliverable 1.2 are listed in Appendix A and a full list of the requirements with comments and cross reference to these challenges can be found in Appendix B.

5.1 Overall

The devices running the MIDAS middleware are usually small and highly portable. This means that they are easily lost or stolen and might end up in the wrong hands. Everything stored on the device might then be modified, including stored message or the MIDAS runtime files themselves.

SR 1 **(M)** Users must be able to report lost or stolen devices at which point they would be considered untrusted by the network. Reports should be impossible to deny or withdraw. Special care to prevent denial of services has to be taken.

SR 2 **(M)** Data stored on nodes should not be easily available if device is stolen or lost.

SR 3 **(M)** Users should not be able to elevate their permissions by changing files locally on device. This includes modifying the middleware software itself.

SR 4 **(M)** The devices should verify the identity of the user (user authentication) so that stolen devices are rendered useless.

SR 5 **(M)** General remote attacks on nodes should not reveal sensitive information to an attacker.

MIDAS provides services for the masses. This means that it has to be user friendly and require little technical insight. The users might also in some cases be in a stressful situation where usability becomes even more important.

SR 6 (H) Security solutions in MIDAS should be easy to set up, both to allow rapid network deployment and eliminate the need for technical personnel.

SR 7 (H) Authentication mechanisms must be simple to use for non-technical users.

SR 8 (M) Tools for managing permissions must be intuitive and self explanatory

SR 9 (M) Most security settings should be managed with little or no user involvement.

5.2 Access control

The purpose of the MIDAS network is to enable nodes to share information without any prior existing infrastructure. Some of this information is gathered and produced in and by the network (e.g. sensors), and some might be stored locally on nodes before connecting to the network. Either way some of this information is sensitive and privacy has to be protected. This includes who should be able to add, read, modify and delete information from the MDS.

SR 10 (H) Sensitive information must be protected when stored and communicated.

SR 11 (H) There must be limitations as to who gets access to what information.

SR 12 (H) Access granting should be possible both off-line and on-site.

SR 13 (M) The middleware must support well defined roles.

SR 14 (L) Access should be granted to a given role or user. Roles might change dynamically as users leave or arrive.

One of the functional requirements of MIDAS is to depend on a central node. Without a central authority access control has to be enforced by every node sharing information (e.g. storage nodes) and be evaluated according to well defined rules shared by every node. Information providers should be able to determine who will be allowed access to the data they generate.

SR 15 (M) Every node should be able to evaluate access permissions and the result the of evaluation must be the same on all nodes. This includes authentication, role and permission verification.

SR 16 (M) Users (or applications) providing information into the network should be able to set access restrictions on the data they provide.

SR 17 (M) Access restrictions should in some cases be possible to bypass.

5.3 Routing

Nodes in MIDAS should be able to setup an ad-hoc network without any previous knowledge of each other. Trust relationships between nodes can be established prior to network creation or on the fly, but this should not be necessary for the network to form.

SR 18 (H) Nodes must be able to participate in the network without access credentials having to be established beforehand.

SR 19 (H) Untrusted nodes should be able to handle (store, route) sensitive information.

Protecting the middleware at the network layer is complicated by the openness of wireless communication. Making it difficult for malicious users to do real damage is important in these open platforms. A malicious node in this setting is a node that intentionally tries to attack/disrupt the network.

SR 20 (M) MIDAS should protect packets from being modified in transit. Malicious nodes should not be able to disrupt service, gather information, send false messages or corrupt the routing protocol (includes dropping, replaying or injecting packets).

SR 21 (L) Malicious nodes trying to disrupt the service should be detected and locked out. This has to be done in a way so that obvious denial-of-service attacks are not possible.

SR 22 (L) Malicious nodes should not be able to gather sensitive information when routing messages.

5.4 Communication

SR 23 (H) MIDAS must be able to unambiguously determine the source of a message.

SR 24 (M) The trust model between peers in the MIDAS network must support ad-hoc setup and match any hierarchical structure.

SR 25 (L) Some messages should force returning a receipt upon receiving/reading.

In service domains where security is crucial, trust relationship prior to network creation should be taken advantage of so that participants can communicate securely.

SR 26 (M) MIDAS needs to support secure communication between nodes trusting each other without having to trust every routing node in between. This includes message integrity, data protection, non-repudiation and protection against replay attacks.

5.5 Ethical

The ethical requirements cover general properties of the middleware. They state the intention the middleware should have.

SR 27 (H) Sensitive information (as seen from the system) must be secured to prevent unwanted exposure.

SR 28 (H) The middleware must protect sensitive information from unauthorized access.

SR 29 (H) The middleware must protect the integrity of information.

SR 30 (M) Some context information is considered sensitive and should therefore be protected. This includes aggregated data.

SR 31 (M) Usage of services and peer interaction is sensitive information and must therefore be protected.

SR 32 (H) Personal information should be deleted at the request of data owner.

SR 33 (M) Users have to be able to see what information they share to others.

SR 34 (H) Users should be warned about what the information they give will be used for.

SR 35 (M) Default settings must favour a high degree of privacy.

Chapter 6

Security building blocks

This chapter describes in detail the security building blocks of our proposed solution and what they do for the overall security in MIDAS. Figure 6.1 show the original architecture from Section 3.3 and where security components fit in. The new components are Signed context certificates, Key server, Row level integrity, Access control, Signed permission certificates and Cryptographic libraries.

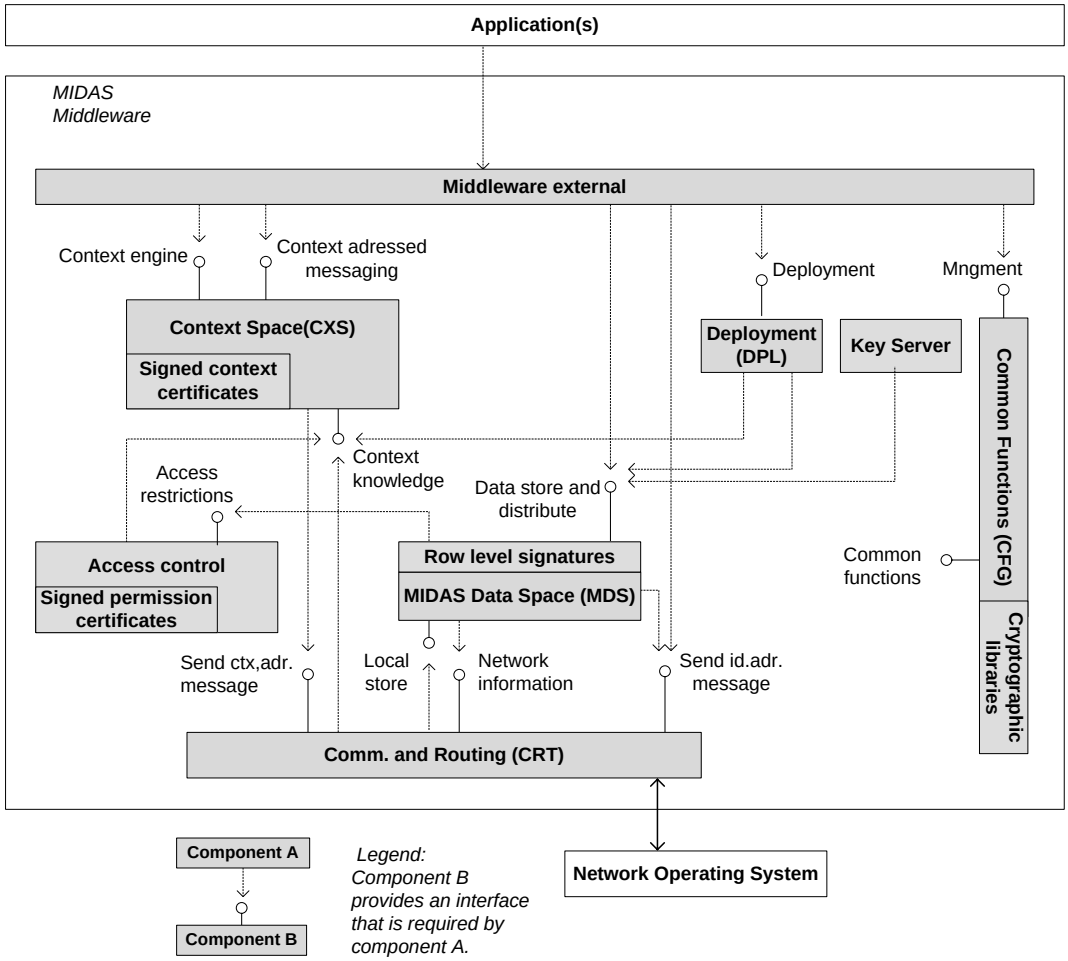

Figure 6.1: Architecture overview with security building blocks.

Row level signatures One of the key element of the secure architecure is the row level signatures (Section 6.2.2) which proves both data origin and integrity. This is used to secure MDS which is a central part of the MIDAS middleware. The other components rely on a secure MDS for storing and retrieveing data.

Key server The key server stores public keyrings in MDS so they are available in the entire network. The component would also implement routines for looking up keys and finding key chains (described in detail in Section 6.1.3).

Access Control The Access control module (Section 6.3) verifies access permissions for MDS and other components. By decoupling MDS from Access Control (AC) we get a clean separation of concerns between the two components and the AC module can be used by other components as well. Signed permission certificates (Section 6.3.2) are an integral part of this component.

Common Functions The CFG component provides common functionality to all the other components and cryptographic library functions needed to implement the security solution will be placed here. This is not in itself a security building block so will not be described here, but is discussed in Section 7.5. Typical functions are message signing, message encryption and signatures verification.

Context Space Signed context certificates (Section 6.4) are not a standalone component, but play an important part within the context engine to provide trusted context that can be the basis for access control.

We will in this chapter first define a trust model that is needed to realise some the security requirements in MIDAS and base our choice of technology on this. We then look at different ways to utilise this technology to cover the security requirements and implement the new components of the architecture.

The requirements referred to in this section can be found in Appendix B and are the same as the ones described in Chapter 5.

6.1 Trust model

A trust model as defined by Stinson et al. [14, p. 2], describes how entities in a system can cooperate by looking at the confidence level entities grant each other. In the real world we might trust people we know, have met before or who have been recommended through common friends. Capra argue that this should be the basis for a distributed trust model [6]. He also suggest that trust is individual and a trust model not allowing users to define their own trust relations are to restrictive. In some MIDAS service domains the communication in mostly public and the security requirements are low, whereas others need a high degree of security. Here trust is crucial and has to be established before the network can be created. Trust should therefore be possible to establish both in advance and ad-hoc.

6.1.1 Different types of trust models

Two approaches to trust modelling have been identified by Choi et al. [8]:

- **Reputation based**, where trust is calculated by looking at transactions (negative and positive) performed by nodes.

- **Trust expressed between users or entities**, where users actively established trust between peers or use a trusted third party to enable trust.

A reputation based system can be automated. However, without a third party to maintain trust levels, every node has to calculate its own trust parameters for each node it needs to interact with. Since the MIDAS network has a short lifetime there might be too little data to base this calculation on.

Trust expressed between users or entities is the approach used in personal X.509 certificates [20] and Pretty Good Privacy (PGP) [49]. Both use a Public Key Infrastructure (PKI). The two most common PKI models are the centralized hierarchical PKI [24] systems and the web-of-trust PKI systems [11].

X.509 is based on the centralised hierarchy model where the root CA certificate is well known and trusted by all participants. The benefit of this model is the ease of finding a certificate path. A path exists if one or more certificates higher in the hierarchy can be used to generate a link between two certificates. This can be seen to the left in Figure 6.2. Since certificates can be signed by one and only one CA higher in the hierarchy the complete path can be enclosed within a signed message.

The centralized approach is proposed in the work done by Puzar et al. described in Section 3.7.2. X.509 is widely supported and the standard in most PKI systems. The centralised model is however in conflict with a fully distributed system and should be avoided in MIDAS to get the full benefit of an ad-hoc distributed network. The most prominent problem is that users cannot control their own trust relationships and ad-hoc setup is difficult.

PGP [61] is a public key cryptography scheme based on the web-of-trust model. PGP is fully distributed where users sign each others keys. Signed keys are then stored on a keyring accompanied by the public key. The web-of-trust model in PGP can be used to generate an hierarchical structure if needed, but its normal operation is cross-certification, like the X.509 cross-certification mesh. Using the web-of-trust model in MANETs has been proposed by Ngai and Lyu [7] who discuss a fully distributed authentication mechanism based on PGP and natural clustering in mobile networks. The cross-certification in PGP quickly grows out of control if every user has to sign every other user key (especially since signing should be done by meeting in person). Trust in PGP can therefore follow chains of certificate from one user to another[1].

Figure 6.2 show the difference between the two models. We see here that a web-of-trust can express more complex structures, but can grow out of proportion because of the cross signatures. In the hierarchical model both certificate #1 and #2 can trust each other via the same chain (by just following the lines in the figure). In the web-of-trust model, node #3 and #4 have to use different chains to gain mutual trust because trust relationships are directed (following the arrows in the figure).

[1]This is a bit simplified. Keys in PGP are assigned a trust level that determine whether a chain can be followed or not. This is however less relevant in this discussion

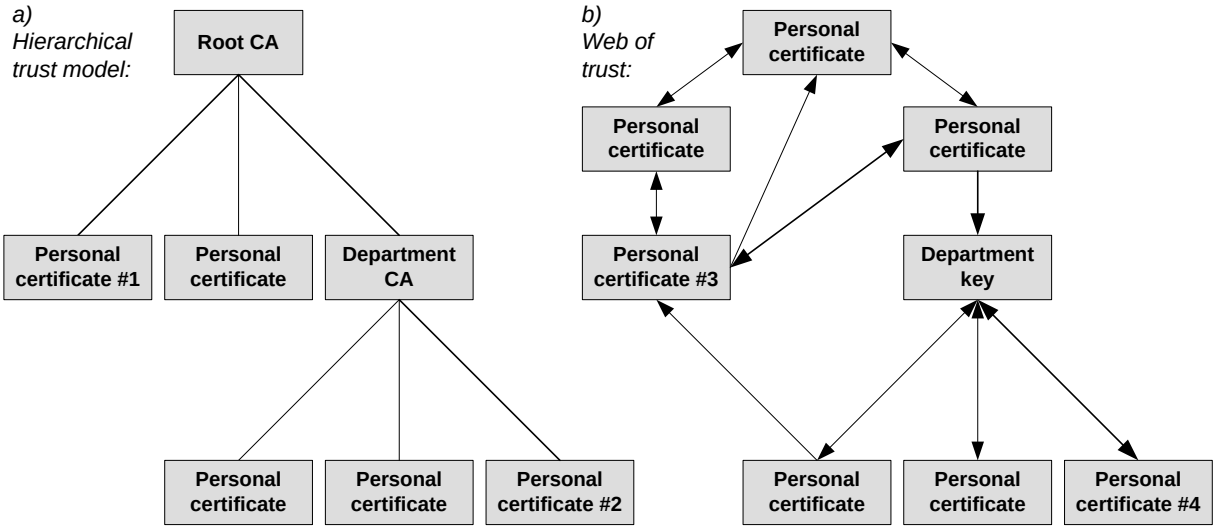

Figure 6.2: Different trust models.

6.1.2 Trust model in MIDAS

We propose to adapt PGP and the web-of-trust model where peers can both verify and establish trust between each other on the fly. A strict hierarchical structure can be modeled on top of the web-of-trust (as seen to the right in Figure 6.2) in for instance the emergency scenario if key initiation is possible before the network is created. The added benefit is that one or more hierarchies can be linked together using cross-verification.

PGP uses a key server where public keyrings can be stored. This is a central repository for finding public keys and certificate chains. The MDS can be used as a simple key server by adding a table for holding keyrings. The global nature of MDS makes sure that everyone will have access to search and download public keys. The keyring of a user should be uploaded to the key server when the middleware starts up and updated whenever changes occur (e.g. when new keys are signed). As long as MDS is able to synchronise all storage nodes, this ensures that trust relations in the network are available to everyone and up to date. This will also ensure that message signatures can be verified after the signing node has left the network.

In Section 6.3.1 we describe a way to make sure that uploaded keyrings can only be updated or deleted by the owner of the key. PGP keyrings are meant to be public and does not require any read restrictions.

The trust model help to achieve Requirement SR 15 by defining how trust should be evaluated in the network. The use of PGP solves the problem of unambiguously determining the source of a message (Requirement SR 23).

6.1.3 Building a web-of-trust

An inherit problem with PGP is that the cross certification quickly grows out of proportion and chains of trust are introduced to reduce this problem. Still, a certain

amount of signed certificates need to be available for chains to be found.

PGP key signing parties [61] are the normal way for PGP users to expand their web-of-trust. People having to meet in advance is however not desirable in MIDAS. Instead nodes could utilise the concept of multichannel security [58] to sign keys fast and secure. With multichannel security, public keys can be exchanged using infrared links or another short range channel whenever peers are in direct reach of eachother [4]. A confirmation of the identity of the person can be done before the key is signed. In this way new security associations can be establish in a self-organized, distributed fashion. Notice that the identity claimed in the PGP key has to be verified, since there is no way for PGP to enforce this on its own.

To create hierarchical structures PGP keys have to be created that are not personal, but form nodes in a tree (as seen to the right in Figure 6.2). Personal keys are then signed as children of this node. In the emergency scenario we would model a police department, district and unit. All police officer keys should trust their units key. This constructed tree structure and the ad-hoc nature of PGP solves Requirement SR 24.

6.2 Secure data storage

The components in MIDAS, including CRT, CXS and the DPL use the MDS component for normal operations, storing or retrieving data. If applications cannot store data without some form of ownership or guarantee of integrity and availability, then all applications will be vulnerable to injection attacks as well as information stealing.

6.2.1 Data protection

Protecting MDS thus becomes an important part of securing MIDAS. MDS as described in the MIDAS documentation [36] has no access control nor notion of ownership[2]. All data in the MDS can be modified by any node and all can contribute data. In a fully distributed system the storage nodes should be selected based on storage capabilities and position in the network, to optimize availability. This mean that storage nodes usually are untrusted. Since local access restrictions are possible to bypass, just by accessing the storage medium, special care has to be taken to prevent this from becomming a problem. A malicious storage node should not be able to modify data in the entire storage network.

What is needed is data integrity to verify that data has not been corrupted, data origin to verify source and data confidentiality to protect sensitive or private data.

6.2.2 Row level signatures

By signing a database operation and storing the signature in the row together with the data for later verification we introduce the notion of users into the system. Every row in MDS stores the node id that performed the operation. The signature verifies this

[2]The MDS scheme has a *sourceNodeIdentifier* on each insert, delete and update, but this doesn't mean that the row is owned by that node

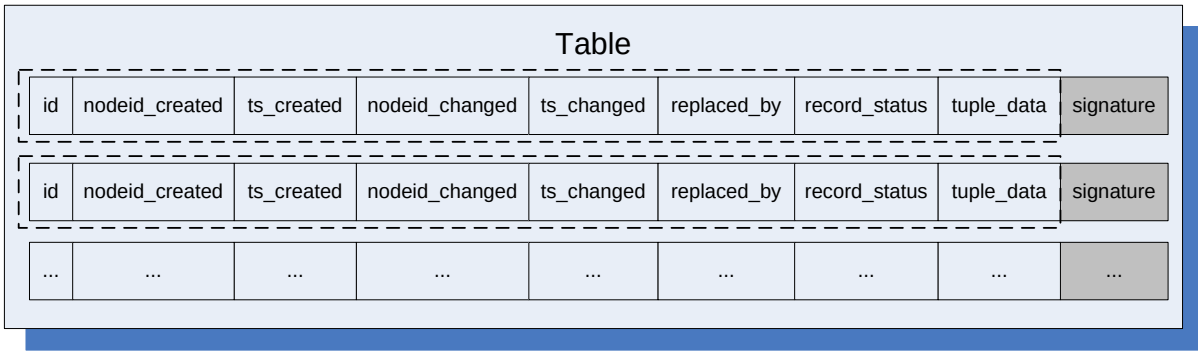

Figure 6.3: Internal MDS table layout with row level signatures.

information and ensures that data has not been tampered with. The signature uniquely identifies the node (Requirement SR 23) and ensures integrity (Requirement SR 29). When data is inserted into the MDS a user will sign the complete row including metadata fields defined for all MDS tables (see Deliverable 2.2 [36, pp. 17-18]). Figure 6.3 shows the table layout with meta data fields. Generating signatures and attaching them to the data row as well as verifying signatures should be implemented by the middleware, so that application developers don't have to think about it.

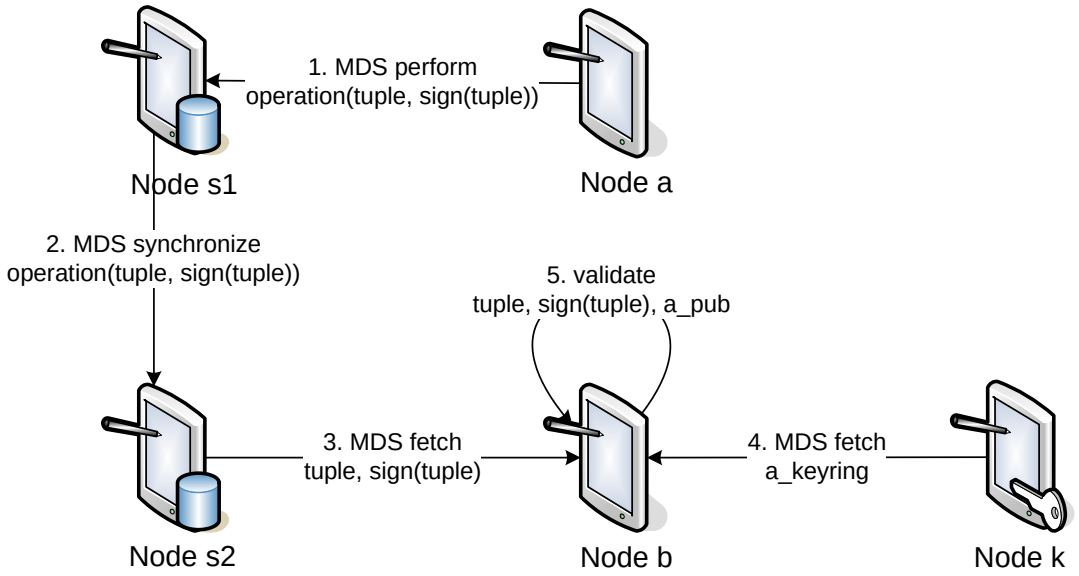

Figure 6.4: Signing database operations against MDS.

Figure 6.4 shows how the communication flows. Node `a` performs a modify operation in MDS (insert, update or delete), and signs this. The operation is performed on node `s1` and replicated to node `s2`. Whether or not signature verification should be performed on these nodes is an implementation issue. Performing validation makes sure that fake rows are ignored at an early stage, at the cost of some extra processing power. Either way it has to be done again by the node reading data because storage nodes are untrusted. In step 3 and 4, node `b` request the data stored by node `a` and

its corresponding key. Notice that node `k` is the key server here, but this role could be performed by both node `s1` and `s2` since the key server really only is a storage node with an instance of the keyring tables. In the last step the row is validated before being accepted on node `b`.

The drawback of these signatures is an overhead in storage space, network usage and computational resources. The first two are proportional to the number of operations to be performed, and constant in relation to the sizes of tuples. The latter overhead is proportional to the size of a tuple because of the hash algorithm which needs to be applied to all the data. Which of these factors will be dominant depends of the size (in bytes) of a typical row. Practical testing is necessary to determine how much performance is influenced by adding this signature.

6.2.3 Row level encryption of data

To protect sensitive data when stored on untrusted nodes (as described in Section 4.1.2), it is necessary to encrypt it. PGP supports an efficient encryption scheme where a symmetric onetime session key is generated to encrypt data [61]. The session key is then encrypted using the PGP public key and attached with the encrypted data. The encrypted data can be stored in the database as binary data. To save computational overhead, only columns containing sensitive data need to be encrypted. The metadata of a row cannot be encrypted as it is needed by the storage node when doing synchronisation.

Adding encryption to MDS will create overhead in storage and require more processing time, as well as increasing battery consumption, which is a limited resource on hand-held devices. Another drawback with encrypting data stored in a SQL database is that the query engine will not be able to operate on encrypted data. Encrypted data has to be stored as a blob and decrypted at the client [43, pp. 8-9]. Clients therefore have to know exactly which data key to look up. The performance impact might also be substantial because of the increased work of data encryption.

Encryption keys have to be managed separately, since storage nodes are considered untrusted. Nodes trying to read the encrypted data will have to request the key from a node trusted to distribute it (e.g. data owner). If access is granted, the session key is encrypted using the requesting nodes public key and the node become trusted to further manage the key.

The result is that sensitive data can be stored in MDS without being compromised, solving Requirement SR 19 by allowing untrusted nodes to store sensitive information and Requirement SR 2, SR 10 and SR 27 by protecting the data if exposed. For some application the overhead will therefore be acceptable.

6.2.4 Database inconsistency

Even if we sign every database operation, malicious storage nodes still have full control over its own database and can delete rows locally, disregarding data changes. These database inconsistencies will only be a problem for the set of nodes using the malicious node as its closest storage node, because the append only database

in MIDAS will not allow these changes to propagate in the network. This threat can be mitigated by querying multiple storage nodes and comparing the results, but that would add additional overhead to the system. The relaxed database consistency demands in MIDAS allows for this kind of inconsistency when network partitions appear or are merged, so the problem is not uncommon. The MDS synchronisation feature should be able to handle this.

6.2.5 Storage media data protection

Data can be corrupted on the storage medium locally on a node and checksums should be used to detect these corruptions. When the middleware is starting up or if lost or stolen devices are retrieved, these checksums would validate the integrity of data. The checksums do however need to be signed to prevent malicious modification since an attacker would be able to regenerate valid checksums. A hierarchy of checksums could be used to minimize the job needed when data is inserted or updated. This is what is done in git's object database [17].

If sensitive information has to be protected on lost or stolen devices (Requirement SR 2), then the data has to be encrypted. This would mean that a symmetric session key should be used to encrypt the entire database and the session key protected using the private key in the node. Research done on encrypted file systems indicate that the overhead will be manageable [32], however battery consumption and slow CPUs are not taken into account.

Both of these schemes might be to expensive with regards to performance to be implemented at all and it may not even be deemed necessary to protect against these kinds of risks. Both schemes also require that the private key in the node isn't compromised when control of the device is lost. Solving this will introduce more inconveniences to the users and possibly new security challenges.

6.3 Access control

Without access control in MDS we can envision scenarios where malicious nodes can inject, change or delete[3] data to stop programs from functioning properly. This can be done by writing own applications using the API directly or by maliciously modifying the middleware software itself. Data can also be modified during data synchronisation between storage nodes. What is needed is an access control mechanism to determine what operations (insert, read, update, delete) a node is able to perform.

In Section 6.2.2 we described a way of signing database rows to be able to uniquely identify which user performed a given database operation. With this notion of users, implementing an access control module becomes possible. Here we introduce three schemes with different properties that can be combined to support a both efficient and fine-grained access control implementation. Both presented schemes are based on the trust network and the use of PGP keys. Users will therefore not be able to elevate permission by changing files locally, solving Requirement SR 3.

[3]Data can only be flagged as deleted in MIDAS, since the database is append only. This is still a problem since we cannot verify who deleted the row

MIDAS services have different needs for access control. Seitz and Pierson looks at different access control models [1] and what problem they solve.

- Discretionary access control
- Mandatory access control
- Role based access control

Discretionary access control (DAC) says that the owner of an object decides who has access. The *table policies* is a more strict version of this model. Role based access control (RBAC) grant permissions to roles and assigns roles to users or groups. This is the model used in *row level authorisation*. Both schemes are described later in this section.

6.3.1 Table policies

By introducing a concept of data owner, we can implement a row level authentication and authorisation scheme for MDS. The data owner is the node that initially inserted a row with a given identifier and for some tables only the data owner needs to be able to modify or delete. The most prominent example is tables with context information, where only the node itself should be able to edit its own context information. Since only the data owner is given write access to these rows, MDS nodes can deny requests made by any other node, keeping the data consistent. This can also be used to protect the PGP public keyring (Section 6.1.2). Another example is the DPL module, which stores binary packages in MDS. Only the deployment administrator node should be able to update or modify these packages.

To implement this scheme, rules have to be encoded in the database definition schema, defined for each table. These rules should be maintained by a separate AC module, since the built in database control scheme cannot be used to evaluate these kinds of permissions. The AC module should evaluate whether a given operation should be permitted or not. Figure 6.5 shows how the control flows. When receiving an update request the storage node will check the policy for the given table. If verification is needed, the public keyring of the *sourceNodeIdentifier* is fetched either from local cache or MDS. Then signature is verified, and the operation is either accepted or rejected. Operations with bad signatures will be discarded directly.

The solution is simple and relatively cheap, but is lacking fined-grained control, since only data owner has full access.

6.3.2 Row level authorisation

Seitz et al. describe an access control scheme meant for sharing medical data in a grid environment [48]. This paper describes a very similar environment to MIDAS, where patients own their medical journals and should be able to distribute access permissions for this data. What is proposed is a permission certificate which is signed with the patients private key. The certificate can be validated by anyone who have access to the public key, and the format of the certificate allows for fine-grained configuration. Figure 6.6 shows how such a permission certificate will be generated and a signed

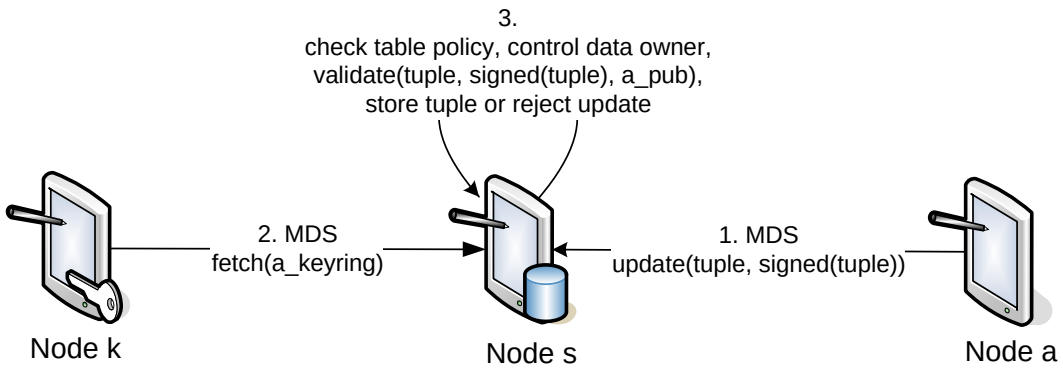

Figure 6.5: Table policy access permission checks.

example can be found in Appendix C. In this solution private/public keys are used to identify users (or data owner). Access is granted to a unique public key.

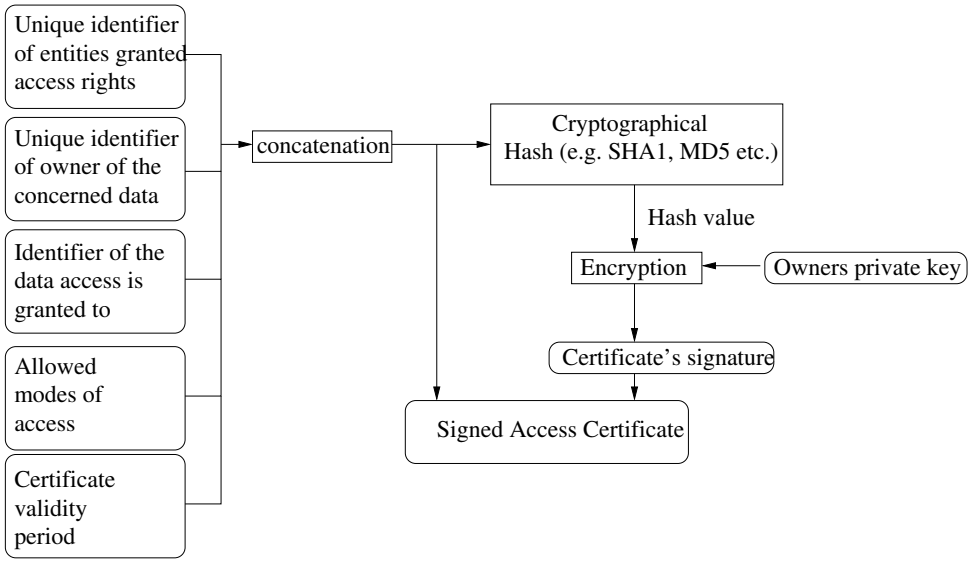

Figure 6.6: Signed permission certificate for row level access control[4].

In MIDAS this can be implemented by having a table for storing permission certificates and using SQL foreign keys to link data rows with permissions. In the grid environment Seitz et al. assumed that the storage servers where trustworthy. This implies that all operations (read, write, update and delete) can be secured via this access control scheme. Since storage nodes are untrusted in MIDAS[5], only update and delete can be controlled this way. Write permission cannot be easily solved with this scheme, but for read control the entire row can be encrypted (as described in Section 6.2.3). The permission certificates can be issued both off-line and on-site and therefore solves Requirement SR 11 and SR 12. They can be set by users or applications, solving Requirement SR 16 and can be granted directly to users or to a role (a given context) solving Requirement SR 13 and SR 14.

[4]Used with permission [48].
[5]Implicitly stated in Requirement SR 19

Since MDS is append only, each primary key will have a set of records depending on how many updates there have been. When an application receive data from MDS it can check the signature attached to the row to verify integrity and identify the node who inserted it. By checking the permission certificate they can further verify that the change was legit. Rows not fulfilling these requirements will be discarded. Since the storage node cannot be trusted, this has to be done by the node itself.

In addition to the storage and computational overhead, this solution introduces a drawback of reasoning about the state of data. The solution requires each node to compare signatures and permission certificates to evaluate whether or not the change is legitimate. Since nodes have different web-of-trust networks and base their evaluation on their own keyring the possibility of nodes evaluating permissions differently can occur, which will lead to inconsistency and conflicts with Requirement SR 15.

It might be possible to solve this by setting some restrictions on who a permission certificate can grant access to. For instance if evaluation could be done from the viewpoint of the owners keyring, the evaluation would be disambiguous. We will not look into how this can be done here.

6.3.3 Row level encryption and session keys

Since any node in MIDAS can take the role as storage node there will inevitably be situations where the storage node cannot be trusted. Since any node has full access to local resources, sensitive data has to be encrypted if privacy is to be protected. Seitz et al. [1] have described a solution where data is stored in a grid environment where storage nodes in the grid are untrusted. The solution uses private/public keys like in our proposed solution, encrypt data on a row basis using a session key and introduce a trusted key server for managing access to these session keys. This is very similar to what we want to do in MIDAS where data should be stored globally without being available to everyone, including the nodes it is stored on.

The method proposed by Seitz et al. is designed to operate with any access control scheme so it will work with the permission certificates described earlier. As described in Section 6.2.3 all tuple data of a row will have to be encrypted, with the implications described there. In the MIDAS network, if no trusted key server is available then the data owner can himself act as a key server. In this way the system will continue to work in a fully distributed manner, but can take advantage of supporting infrastructure where available. By using session keys stored separate from the data we solve Requirements SR 10 and SR 19 which states that untrusted nodes should be able to handle (store or route) sensitive data. We also protect sensitive information from unauthorized access (Requirement SR 28).

6.3.4 Where to evaluate Access Control

We have already seen that access permissions should be verified by the requesting nodes because of untrusted storage nodes. However if no validation is done at the storage nodes, garbage data might be inserted by malicious nodes. To prevent this, storage nodes could do the same evaluation and deny operations that do not validate.

This leads to a more consistent database and less work for the user nodes. The cost of this is slower modify operations since more processing has to be done.

If every node evaluates operations themselves then the result of evaluation will depend on the network of trusted peers for any given user. This will result in inconsistencies between nodes as they trust different sets of nodes. This is OK in MIDAS, since database consistency is relaxed as it is. If however, evaluation is to be done at the storage nodes, then the result has to be the same as if it was evaluated by the data owner. That is, no ambiguities must be allowed.

If only the data owner should be able to modify data, the evaluation is straight forward. The signature is simply verified using the key associated with the source node. However, when permission certificates are used, permission will have to be evaluated from a certain perspective (e.g. a given public keyring).

6.3.5 Central nodes

Even though MIDAS should be able to operate completely ad-hoc, there is a need for central nodes that should be taken advantage of if trusted. The central node can take on the responsibility of validating permissions and distributing session keys, taking the workload off small and mobile devices.

If an completely trusted central node also acts as a storage node then the work of verifying signatures can be placed here, relieving the user nodes of a lot of overhead.

6.4 Trusted context information

In the current version of MIDAS, context is fully controlled by the user and without signed records it would be possible for one node to set the context of another. In a closed network where all participants are known (and trusted) in advance this approach might work. This is an assumed precondition in the solution by Puzar, for the emergency scenario described in Section 3.7.2. We want to eliminate this assumption in our solution.

With the mechanism of signing database operations as described in Section 6.2.2 we ensure that users can no longer set context for others. However, in our fully distributed network, users still have full control of their own context. We propose to specify a signed *context certificate* adapting the approach of signed permission certificates described by Seitz et al. [48] introduced in Section 6.3.2. This context certificate should state what context is set by the certificate, for which node it applies, validity period and signing authority. Figure 6.7 shows how a context certificate would be created.

By checking the signature of the context certificate it would be possible to verify the entire certificate. If the key used is trusted we can also trust the context set by this certificate. The signing authority does not have to be the same as owner. For instance in the emergency scenario a master key could be generated for each local police unit or department. This key would be used to issue context certificates in advance and would be trusted by every policy officer in that department. Figure 6.8 shows how a hierarchy of keys and certificates can be created for the emergency scenario. Here a

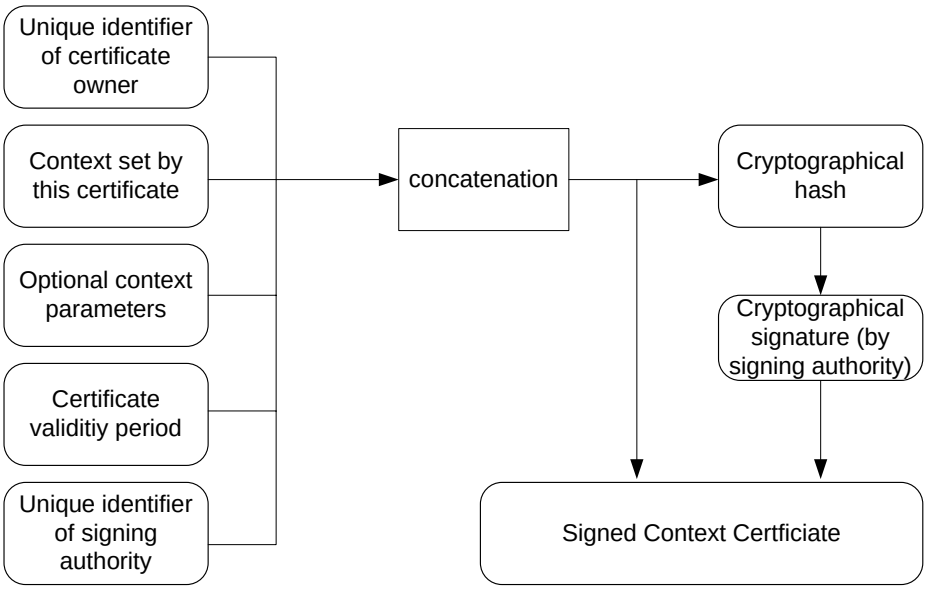

Figure 6.7: Signed context certificate[6].

police officer hold both a PGP key signed by the department master key and a context certificate signed by the same key. The department key can be used to sign other keys to extend the trust network.

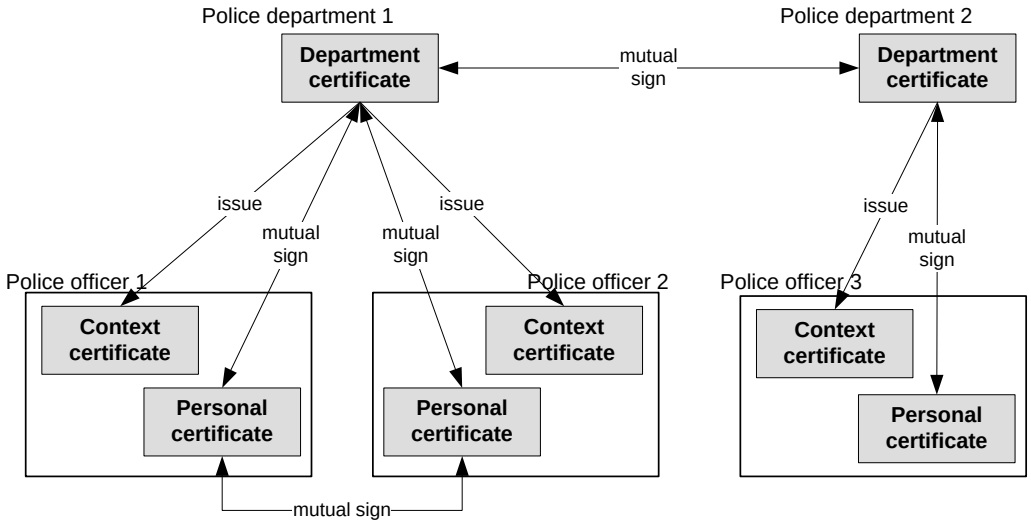

Figure 6.8: Signed context certificates in a constructed hierarchy.

The certificates can be used to define roles in a RBAC system. Combined with permission certificates described in Section 6.3.2 they help to solve Requirements SR 11, SR 12, SR 13 and SR 14.

This model has great flexibility, but create additional overhead for context querying, if context has to be verified. For some context this overhead is acceptable (verifying

[6]Based on Seitz [48]

the context *Police officer* makes it possible to send sensitive information). For other context information (e.g. *Having breakfast, unavailable for discussions*) this might not be needed. Context information derived from sensor information might change so often that creating a context certificate for each new state might not even be feasible. Because of these issues, signed context certificates must be optional and verification enforced in the application, rather than the middleware.

6.5 Secure communication

Since MIDAS uses wireless links for communication it is possible to both eavesdrop, modify and insert traffic into the network. This means that traffic should be protected in transit. Security primitives needed are data integrity, data confidentiality, data origin authentication, non-repudiation and protection against replay attacks.

6.5.1 Network packets

On the network layer there is a need for data integrity. This can be done using a simple hash-function (like sha-1) to checksum packets. Checksums are relatively cheap both to generate and verify. However they do not protect against maliciously modified packets, only random data corruption. Packets could instead be signed using the PGP key to solve this. The signature both verify the content of the packet and which node sent it. Nodes forwarding these packets does not necessary need to trust the key with which the packet was signed. Knowing only that the same key signed all packets and that they where not modified during transport would still increase the overall security. Packets without a valid signature should be dropped.

All packets in MIDAS also have a timestamp and a TTL. Replay attacks can be thwarted by storing packet signatures for the specified TTL. Messages matching a stored signature can be discarded. Signatures should be checked on every node in its path toward the receiver. By discarding false packets as early as possible bandwidth can be saved at the expense of processing power. As we will see in Section 6.7.1, this also makes it possible to detect malicious nodes. The use of signatures and TTL solves replaying, injecting or modifying packets from Requirement SR 20.

6.5.2 Routing protocol messages

MIDAS nodes exchange routing information at regular intervals to detect new nodes and update routing information for mobile devices moving around. The information is used to keep an up to date routing table on each node. Protecting the state of the routing protocol from malicious modification is important for the network to operate efficiently, but difficult to do in practice.

When receiving routing messages a node will want to verify the sender of the message. This means that all routing messages should be signed with the private key of the node, making it harder to attack the routing protocol state by injecting routing packets (Requirement SR 20). Echo packets and heartbeat protocols can be used to discover

outdated or erroneous routing state.

6.5.3 Application messages

The MIDAS middleware routes application messages between nodes. These messages might contain a request for work to be done or simply contain a human readable message and should be signed to verify data origin and integrity (Requirement SR 23 and SR 29). When auditing records are generated the signature should be stored with the messages for later verification. By signing a message we also achieve non-repudiation (from Requirement SR 26). This means that when a message is sent, the sender can no longer claim that the message was not sent by him.

Data does not need to be signed both at the network and middleware layer so a mechanism for applications to choose only one would save bandwidth and processing power.

Message confidentiality is used to make sure that only the intended receiver of a message is able to read its content, which is needed for Requirement SR 10 and SR 27. This is done using the encryption scheme in PGP as described in Section 6.2.3. Sending encrypted messages is rather expensive, both with regards to processing power and increased use of bandwidth. Only messages containing sensitive information should be encrypted.

6.5.4 Securing context addressable messages

When sending CAM the address of all recipients might not be known and obtaining the public key for these recipients becomes impossible. Since the public key is not available, the method described in Section 6.2.3 of encrypting the session key cannot be used. Nodes receiving encrypted CAM messages would have to actively request the session key.

A novel method, giving full control over who gets access to the key, would be to send these requests back to the message origin. This however, does not scale well with a lot of recipients. Spreading the load of re-encrypting the session key on the nodes that has already received it would scale much better. It then has to be assumed that nodes receiving the key is also trusted enough to redistribute it. Figure 6.9 shows how the load of encrypting the session key can be distributed on the nodes receiving the message. Each node encrypt the session key with n ($n = 2$ in the figure) different public keys in each phase. This scale well in theory. In MIDAS the topology of the network has to be taken into account as well, but this is out of scope for this thesis.

6.5.5 Message received verification

For some applications there is a need for the sender to verify that a message is received and read (Requirement SR 25). The recipient must sign an acknowledgment upon opening and send it back to the original sender. This approach is described in RFC2634 [19] and by Coffey and Saidha [9]. A problem with this simple approach is enforcing the recipient to reply. The sender would need a timeout, after which

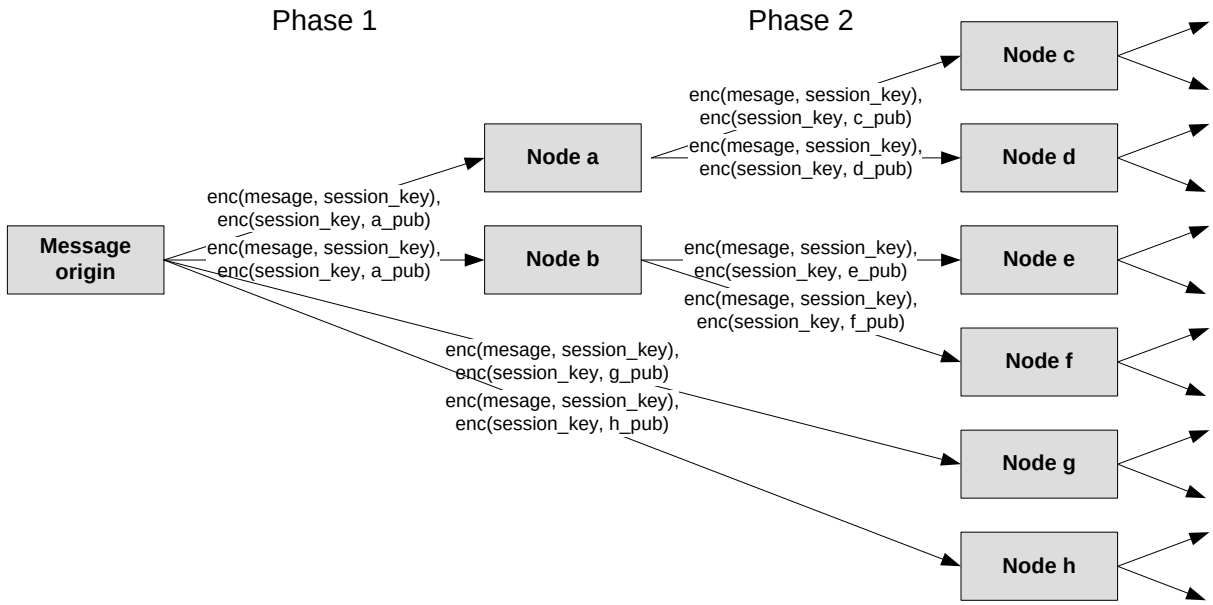

Figure 6.9: Distributing the load of sending encrypted CAM messages.

it will resend the message assuming it was lost. If no receipt is received than the sender will have to assume that the message did not reach its destination. This can happen since MIDAS makes no guaranties as to message delivery. Coffey and Saidha describe a more elaborate system involving a trusted third party, where recipients have to acknowledge fetching the message, solving the problem of forcing the recipient to reply. As this require a trusted third party, it is not suitable for MIDAS.

If this security primitive is needed it should be implemented at the application layer as it would not be needed in most service domains.

6.6 Authentication

Authenticating a user is important to prevent lost or stolen devices to be used for mischief. Authentication has to be done on different levels to service different needs.

We suggest a solution where when a node connects to the network it will prompt for a private key to use for communicating securely. By presenting a valid private key the user authenticates towards the **device**. The devices will recognise the user based on the fact that he has the private key and knows the pass phrase. User-to-device authentication is required by SR 4. If the user does not already have a private key one will be generated by the middleware. This key will have no established trust in the network so it will only give access to public information. Since users can generate a key on the fly, Requirement SR 18 is fulfilled.

When you use a private key to send messages and retrieve data you authenticate to the **network**. You sign your request with your private key and the network will validate this signature and grant access based on your web-of-trust relations. With a generated key all restricted access requests will be denied.

Middleware to middleware authentication or node **peer** authentication can be done when one node wants to authenticate another node in the network. This could be appropriate for instance before initiating MDS data synchronisation. A signed timestamp or a protocol based on a challenge/response scheme with nonces can be used to confirm that the private key is available to the other node.

Last, if a high level of security is needed, the device should authenticate itself towards the **user**. That is, the device should confirm to the user that the hardware and software has not been tampered with and can be trusted. This is probably the hardest problem to fix and many solutions has been suggested for what is called a trusted computing platform [40]. Most solutions require some tamper proof hardware to be placed in the device that can do validation before the software starts. A simpler solution would be to use a read only medium for storing the software, but this would not be as secure.

6.7 Misbehaving nodes

Nodes might misbehave due to misconfiguration, virus infection or when operated by a malicious user. Either way we want to limit the amount of damage that can be done, while not opening up for new security weaknesses like allowing for easy denial of service. In any given scenario the number of malicious nodes will presumably be low. If a large amount of malicious nodes (a significant percentage) exist, we probably won't be able to protect the network anyway. All we can do in this situation is to protect sensitive information from leaking.

Problems that occur when trying to protect the network and the routing protocol include:

- Detecting nodes that misbehave
- Limiting the damage done by misbehaving nodes.
- Making sure that DoS attacks, by getting nodes blocked, is not possible

6.7.1 Detecting misbehaving nodes

If nodes is programmed to check the signature and TTL of received packets, corrupt or malicious packets are dropped as soon as possible. Additionally we can draw some conclusions whenever we receive a packet with a bad signature: Either the packet was corrupted during its last transmission, or the previous node did not behave as it should have done, which was to drop the packet.

A certain amount of faulty signatures must be allowed due to random corruption by the wireless link, but malicious nodes could be identified this way (solving the detection part of SR 21). More research needs to be done to verify how effective this approach might be.

6.7.2 Handle misbehaving nodes

When misbehaving nodes are detected the middleware has to determine what to do with them. Blocking out nodes is undesirable because of the danger of detecting false positives that block out legitimate users. A novice approach would be to send a message to the user of a misbehaving node. This scheme has been tested by the police in Amsterdam where SMS messages are sent to stolen phones on a regular basis [56, p. 27]. The messages act as a deterrent, making the phones difficult to resell. Unsuspecting MIDAS users will in this fashion become aware that their device might need reconfiguring. Malicious users that tries to disrupt services will receive what can only be considered as spam which may act as a deterrent.

Another approach is to use the features of the routing protocol to adjust the cost of routing via the misbehaving node. By routing via other nodes whenever this is possible, the amount of damage that can be done by malicious nodes is limited. This partially solves Requirement SR 21. The possibilities of harming legit nodes are small and malicious nodes are handled, but not locked out. Packets coming from a malicious node can still be accepted if the signature can be verified.

By allowing a malicious node to stay in the network we are still vulnerable to DoS attacks where a node flood the network with false signatures that have to be verified before they can be discarded. This is slightly worse than nodes trying to jam the network frequency, which is always possible, even if nodes are blocked out [51]. This will however only affect nodes in the immediate proximity of the misbehaving node.

6.8 Node identification

In the MIDAS middleware it is critical that node identifiers are unique. Each node is also associated with a number of network endpoints [39, pp. 14-17] that is used to address a given node based on the selected network technology. Mapping services maintain information on how to look up endpoints from node identifiers and back. If PGP is used as a cryptographically secure way of identifying a node an additional mapping is needed between node id and public key.

To avoid this additional mapping, the fingerprint of the public key [61] can be used as node identifier. The fingerprint is a 42 character excerpt of the public key and the chance of collision with other keys is relatively small. The fingerprint is available when the middleware is booting which is a requirement in the current version of the MIDAS middleware [39, pp. 14] and fits in the reserved 64 character field.

Since all nodes should upload their public key to MDS when they connect to the network, the PGP real name can be used as an human readable mapping between names and node ids. (NB: Notice how PGP Real name are not necessarily unique.)

The node id will be set when the middleware starts by reading the keypair given. If no key is provided then one will be generated.

6.9 Certificate revocation

When using private/public key cryptography it is often an issue with public key (or certificate) revocation. The need exists because private keys get lost or are otherwise compromised and the owner of the key would like to revoke the use of the corresponding public key. For PGP this is done by issuing a revocation certificate that needs to be distributed to all users of the public key. Usually this means updating your public key on the key server.

For MIDAS the revocation certificate should be uploaded to MDS and the middleware should check for revocation certificates at regular intervals. Revocation certificates solves Requirement SR 1.

Chapter 7

Different configurations

This chapter shows how the building blocks from Chapter 6 can be put together to achieve different levels of security and which security requirements are solved by each configuration. The four configurations are: Baseline, Pre-installed certificates, Signing database operations and Signed context- and permission certificates. The configurations are mentioned in an increasing level of security.

7.1 Baseline

The MIDAS middleware sets up a distributed, ad-hoc network and in Section 6.1 we argue that to enable security in this network the web-of-trust model should be used to establish trust between peers. This is implemented using private/public key cryptography and cross certificate signing. We propose to use PGP, but X.509 certificates can also be used. Every node needs a PGP key to enter the network and the public keyring should be made available to the network.

This is our minimal configuration which generates the least amount of change to the original architecture. The performance impact is also kept at a minimum. What is needed is PGP libraries available for the middleware, the integration of a key needed to boot the middleware software and tables in MDS to store public keyrings.

With this relatively simple baseline we still have the opportunity to implement some useful security features.

Distributed public key server All nodes upload their public keyring when the middleware is starts up. Storing this keyring in the MDS makes it available for all nodes also after the node has left the network.

Auditing Without some kind of cryptographic signature it is nearly impossible to create an audit log that can be trusted. When messages are signed, this signature can verify that the content of the audit record hasn't been tampered with. By attaching a signature we both identify the node that generated the audit record and ensure the integrity of the stored information.

Unique node identifier The private key used by the middleware can also be the basis for a unique node id. MIDAS uses the node id to map between network

endpoints and nodes. The fingerprint of the private key is an ideal candidate for node id because of the low probability of collisions.

Application level security By having public keys available in MDS and knowing that all nodes have a private key, application developers can add functionality that utilise this to increase the security of the application. For applications that need a high degree of security this means that e.g. full message encryption will be an possibility. Possible application features include:

- Secure messaging system with full encryption.
- Application level access control based on trust network. (Implemented outside the middleware)
- Sender id verification for messages (including CAM).
- Sensitive information can be stored encrypted in MDS.

Application level access control can for instance be utilized by third party applications where services have to be paid for. Paying customers would register their public key when they purchase the service making it possible for the application to verify that the customer actually has paid for the service it receives. Combined with using the private key as the basis for node id this ensure that customers cannot abuse the services by sharing their credentials with friends.

7.2 Pre-installed certificates

The most outspread use of X.509 certificates is to secure web traffic using https to identify websites and encrypt traffic data. This work by having a trust list [27] of approved top-level certificates installed in the Internet browsers. We can imagine using the same technique in MIDAS where the middleware comes pre-installed with a list of approved keys. However, PGP was not intended to be used in this way, so there is no way to automatically accept these keys, they have to be signed manually. When they are, they can be used by for instance the deployment module to verify origin of packages or to distribute official information in the *Sports* scenario.

Packages distribution By having pre-installed certificates bundled with the middleware application packages they can be signed by application developers and verified before installed on the mobile device [1]. This ensures that it will be harder for attackers to insert malware or other malicious application packages in the network.

Official information For the *Sports scenario* there is little need for trust between spectators. They probably don't know each other anyway. What is needed is the possibility to send out official information that can be verified by spectators and therefore be trusted. This is done by distributing a public key in advance for instance on the official website. This key is then used to verify all official messages.

[1] This approach is taken on the Apple iPhone where developers can buy a developer key

7.3 Signed database operations

With signed database operations the system gets integrity checks and data origin in MDS and information there can be validated by the middleware framework. The signature should protect the metadata of a row as well as the actual content and should be used both for MDS synchronisation and when data is being queried by nodes.

Data row integrity By signing database rows the integrity of both the metadata and the actual content is maintained.

Data owner One of the key benefits of signed rows in the database is the introduction of data owner. The metadata of MDS rows in MIDAS already include *sourceNodeIdentifier*, but this field is currently not used to associate data with an owner and can be faked. By signing the row, this field can be verified and table policies be defined.

Table policy access control A simple form of access control can be implemented by allowing only data owner to perform changes to data. This simple scheme can be used to store the PGP public keyring, making MDS an effective PGP key server where keys belong to the key owner and cannot be deleted by others.

Signed context data When using signatures on rows, simple context parameters can be stored without malicious nodes being able to modify them.

Another main advantage of this scheme is that it only introduces minor changes to MDS and does not interfere with how data is being stored or replicated. It is possible to extend the existing MDS component with a module which will handle verification transparent to the application programmer.

7.4 Signed context- and permission certificates

In Section 6.3.2 we described signed permission certificates based on the work done by Seitz et al. [48] and in Section 6.4 we extended this solution to create signed context certificates. Together they form a basis for a fine-grained and flexible RBAC access control system. Roles in RBAC systems can be mapped to context in MIDAS.

Role based access control Data stored in MDS can be associated with a permission certificate stating which user or context is needed to access the data (the certificate also includes mode of access). Users with a signed context certificate will be allowed access to the data. Access is granted when the following restrictions are met:

- The signature of the context certificate (signing authority) can be verified
- The signing authority can be trusted
- The requesting node is verified to be the subject of the context certificate.
- The signature of permission certificate can be verified.

- Permission certificate grants access to context in context certificate.

Using permission- and context certificates impose a significant performance degradation as the principles involved require several messages to be sent (if public keyrings have to be fetched), signatures have to be verified and extra bytes has to be sent over the network and stored.

7.5 Library functions in CFG

We saw that by using PGP keys in all nodes, security features can be implemented by application developers. This includes sending encrypted messages, verifying message signatures, sender identification or even a custom access control scheme. Much of these functions can and should be placed in the CFG library so that application developers do not have to implement these on their own in each of the applications that need it. We do not regard this as a configuration on its own, but rather a description of how we want to make security measures available for developers.

Secure messaging Instead of having to explicitly sign or encrypt messages, application developers should only need to set a level of security needed for any given message and the middleware will take care of fetching public keys, signing, encrypting and verifying.

Encrypting database rows Full data encryption is supported by PGP, so the CFG should provide a way to encrypt sensitive information before storing it in MDS. Access to the data encryption key must be handled by a trusted node (e.g. data owner). The metadata of the node cannot be encrypted, only the actual data of the row. Only rows containing sensitive information should be encrypted since encrypted data cannot be searched by the SQL query engine.

Distributed key server In our baseline configuration MDS acts as an distributed key server by storing public keyrings for all nodes. By signing database operations we also get a limited form of access control for these keyrings. However, a key server should also enable searching for keys based on a number of criteria and finding certificate chains from one certificate to another. This can be implemented as a set of common functions.

Chapter 8

Evaluation and Discussion

This chapter presents an evaluation of the work we have done and a discussion of the results of our solution. The evaluation looks at what we did, how well we think we covered the parts and what we recommends for future work. The discussion justifies and explains some of the decisions taken in earlier chapters. We discuss the overall issues like the value of our research, whether it is feasible to implement the solution, which weaknesses still exist and point out what our solution does better than similar approaches. At the end of the discussion we argue for a reasonable division of responsibility between the middleware and the applications.

8.1 Evaluation of State of the art chapters

The State of the art portion of this thesis required both study of similar systems (peer-to-peer and mobile ad-hoc networks) and previous research of those systems, as well as a comprehensive study of the MIDAS documentation. We focused mainly on the specifics of the architecture and the functionality provided by the middleware when reading the documentation. Many of the individual parts of MIDAS had similarities to other known problems. We could, however, not find any papers where these parts where combined to create a middleware. The main sources we used were IEEE Xplore, ACM Digital Library and Google Scholar. ACM and IEEE represent well established and recognised sources of high quality, and Google Scholar represents the diversity as it combines many sources with good searchability.

8.2 Ethical discussion

The ethical discussion presented in Chapter 4 is an approach to security background, as stated earlier. We wanted to examine what influences security, how information is handled and which factors are critical to success. The ethical ramifications introduced in MANETs have not been as widely researched as other areas of computing. We therefore used an ethical framework from the database/interactive marketing domain where information interaction is in focus. We used the framework to identify potential ethical issues.

With more time and experience we could probably have found more ethical issues and made better ethical requirements. The number of ethical requirements that actually were fulfilled was small, which might be because of their wording or the difficult environment MANETs have.

8.3 Security requirements

The security requirements identified in Chapter 5 are based on concerns expressed in MIDAS Deliverable 1.2 partIII [53][1], ethical issues described in Chapter 4, our security analysis of the architecture in Section 4.3 and common sense when looking at systems needed to handle sensitive information.

However, a more formal approach might have yielded a better result and given a documented process over how security requirements where identified. A reasonable approach would have been to do a risk analysis of the two PoC scenarios in MIDAS and used this to specify requirements. Some of the requirements are also overlapping so a more concise formulation would have been beneficial. A reformulation after the solution was formed would be adapting requirements to the solution, which is a risky approach.

The requirements focus a lot on handling sensitive information and how confidentiality is important. These requirements originate from the emergency scenario. There is however little focus on verifiable context which is at the heart of MIDAS and became an important part of our solution. For future MIDAS scenarios, integrity and verifiability might prove as important as confidentiality and should have been emphasized more in the requirements.

The security requirements where validated by security experts working on the MIDAS project who didn't protest to the formulation, so they are certainly not wrong. We feel that the requirements are both relevant and cover most needs in MIDAS which is more important than achieving a high percentage of covered requirements. Which requirements are met and in which configuration can be found in Table 8.1.

8.4 Security building blocks

A driving force for our security building blocks was the MIDAS requirement of no central authority or trusted third party, and the possibility for fast and ad-hoc setup. We feel we have achieved this using the web-of-trust model which is ideal for this type of network. By choosing to use PGP as the technology to implement the web-of-trust, we were able to solve requirements regarding integrity, data origin and confidentiality both in messaging and for data stored in MDS.

We describe two access control schemes, a simple table policy model ideal to protect data with only one owner, and a more elaborate scheme with permission certificates to define fine-grained access control levels. Both can be used at the same time for different types of data. The simple table policies are part of our own contribution

[1]See Appendix A

whereas the permission certificates are described by other researchers. Both are appropriate for the MIDAS middleware.

The presented building blocks form a complete solution to the security requirements from Chapter 5, however, some areas of the solution have potential for improvements. Specifically the protection of the routing protocol. The solution has not been verified using a proof of concept implementation nor a security assessment. This would have helped proving the effectiveness of the solution. We also assume that some security mechanisms are too expensive, in terms of overhead, to implement. Laboratory research could have been done to verify this claim however time limitations prohibited this.

8.5 Configuration options

The building blocks of Chapter 6 form a complete solution with high degree of security. This level of security might not be desirable for all MIDAS service domains, because of the performance impact and loss of usability. We therefore adapted a modular approach and describe how the building blocks can be put together in different configurations based on specific security needs. This is useful both when security mechanisms are to be incorporated in the MIDAS middleware and for a domain responsible developing a new ontology service model. The different configurations should suit most security needs ranging from no need to a high need.

8.6 Research value

The MIDAS project has design constraints and application requirements not found in any other projects so the overall security solution for this environment combines concepts in new and innovative ways. The requirements that are together forms the uniqueness of MIDAS are: the ad-hoc network model with no central authority, globally shared data space maintaining availability in face of network partitions and the context space allowing reasoning on context data and context addressable. The value of these services would be great in both of the given PoC examples (Section 3.2), but the value is diminishing if the data gathered in the network cannot be trusted.

Our research increases the potential value of the services on the MIDAS middleware by securing them. With the security we have suggested it would be possible to implement commercial services for paying customers without these services being exploited by others. This is an increased value that can be seen in relation to the ethical issues presented in Chapter 4, as a security incident could be expensive for the domain responsible, both in terms of lost revenue and negative media attention.

As MIDAS is a new idea with special properties we have done a lot of work investigating how to best incorporate security features. This includes researching the MIDAS specifics as well as what MIDAS has in common with other systems. The lessons learned for MIDAS can also be used in next generation systems and systems which needs to be fully distributed.

8.7 Feasibility

The time constraints of this thesis did not allow for a PoC implementation. Future work should be done on how the building blocks of Chapter 6 could be implemented in MIDAS. Without any PoC implementation we can only suggest a solution, not go into specific implementation details. However, we will discuss properties of our solution that effect the possibility of an implementation below.

PGP cryptographic library PGP has been chosen as the cryptographic library to generate public/private keys, signatures and encrypted messages. PGP is well suited for the problem at hand because it uses the web-of-trust model that can cross sign public keys and these signatures are stored in a public keyring which can easily be stored in MDS. PGP is also based on well tested technology and the implementation has been scrutinized by security experts from all over the world.

However, it might not necessarily integrate well with the Java platform used in the current version of MIDAS. There exists some attempts at integrating PGP with Java using the Java Native Interface (JNI) and GnuPG Made Easy (GPGME) [18]. When using the native interface the Java code is no longer guarantied to be platform independent and the PGP libraries have to be able to compile and run on the mobile devices where MIDAS will be used.

The fact that the current implementation of MIDAS uses Java, does not mean that this will be true if a commercial implementation is made. Further-more it would be possible to implement a subset of PGP using Java because of the wide range of cryptographic algorithms already available in core libraries. We recommend using PGP as it is the technology best suited for the task, but it might be possible to exchange PGP with other technologies with the same properties if somehow PGP doesn't work with Java. (public/private keys, web-of-trust, message signatures and encryption).

Storage and bandwidth overhead Our solution imposes overhead both in transmitted messages and in MDS. For MDS the overhead of the metadata amounts to 216 bytes (3x64 bytes varchar + 2x4 bytes integer + 2x8 bytes timestamp). By generating a detached signature with PGP the signature takes an additional 65 bytes. The size of metadata will increase with 30%. We think this overhead is acceptable because it is static and won't increase with the size of the messages.

The same overhead is generated with signed application messages and will be acceptable for all but very small messages. When messages are encrypted, PGP first applies compression on the file to make its content look more random. An encrypted message will in many cases use less bytes than the original, due to the compression, and thus save bandwidth.

When signed context certificates and signed permission certificates are used the overhead becomes much larger. The example signed context certificate in Appendix C shows how these certificates can be implemented using XML. The total size will vary, but in our example a human readable version takes 680 bytes. If fine-grained access control is critical for the application this overhead will have to be accepted.

Performance impact In addition to the storage and bandwidth overhead, the proposed security solution has negative performance impacts locally on the node. Cryptographic operations are relatively Central Processor Unit (CPU) intensive of nature and hand-held devices usually have limited processing power. Another problem is the extra battery consumption by the CPU when it is working hard.

The most common cryptographic operation in our solution will be to check signatures. This will have to be done when receiving messages, querying MDS or checking public keys and needs to be fast. We ran a simple test on a 266 MHz Pentium II running in single-user mode (a single process only, no multiuser environment). We signed a 1 kB file containing random data and tried verifying this signatur 1000 times. This should show how much time is needed to simply verify a signature. The result states that verification only took $52.671s$ averaging about 19 verifications a second. Using $52.6ms$ to verify a signature is fast enough for use in MIDAS.

Performance testing is complicated by the fact that the time spent on verification is dependent on the message size. When sending messages we do believe that the bandwidth (especially for bluetooth) will be the bottleneck for throughput, but signature generation and verification will inevitably lead to higher latency. This because the bandwith for bluetooth is about 1-3 Mbit/s[2] depending on the version, and as the testing above showed PGP can verify signatures with a throughput larger than 1Mbyte/s. Without actual testing on the MIDAS middleware it is difficult to determine if this latency will be large enough to become a problem.

When encrypting messages we saw that the size decreased and it saved bandwidth. However, the latency of messages will be even greater than for signed messages because of the added work of performing compression and encryption. If the data to be communicated is sensitive this overhead will have to be accepted.

We suggested in Section 6.5.1 that it would be possible to sign network layer packets, thereby discarding malicious packets early and be able to detect malicious nodes. As MIDAS operates with UDP/IP packets a 65 byte large PGP signature will be unacceptable at this level as the UDP/IP headers are 8 bytes which makes the overhead eight times bigger than the UDP/IP headers. The performance impact from verification at each hop will also seriously increase the latency.

Technological development will help make these problems less of an issue. Powerful hand held devices today typically have a 400MHz CPU.[3] Because of Moores law [45][4] and experience of the past, CPUs will be cheaper and use less power at higher speeds. Battery technology will also increase the operational lifetime of a device.

Architectural changes Our proposed solution follows a modular approach and can be integrated in MIDAS without large architectural changes. Most of our security

[2]According to Bluetooth SIG http://www.bluetooth.com/Bluetooth/Technology/Basics.htm
[3]Based on an investigation of product specifications. E.g. the HTC TyTN II http://www.htc.com/www/product.aspx?id=644
[4]It is viewed as a self-fulfilling prophecy which drives the progress of manufacturers

mechanisms are optional and will therefore only extend the existing API instead of modifying it. The baseline demands that PGP keys are provided or generated on startup and public keys uploaded to the key server. This implies minor changes to the existing logic and database schema. Messages with a signature attached should also automatically be verified which means changing the way messages are received. For database tables where signatures are required this should also happen without developers having to specify it directly. This means modifying the query parser so that it does this automatically. This might prove the most difficult change to implement, but we strongly believe that its possible.

Overall we think that our solution is feasible and that MIDAS will be able to integrate our proposed security solution smoothly.

8.8 Design consequences

In this section we discuss benefits, pitfalls and unresolved issues in our design. We highlight why we have designed the solution the way we did and what problems the design doesn't solve. To conclude we present the issues we think is important for MIDAS to examine in order to be confident that security is managed well.

8.8.1 Benefits of the proposed design

The benefits of our proposed solution is first and foremost integrity. With the web-of-trust and personal keys, data can have an owner which is identifiable. This means that our solution at least strengthens auditing which is important when investigating an emergency situation after the incident. Integrity is made available in the baseline configuration presented in Section 7.1.

The second benefit of our solution as we see it is close integration with the rest of MIDAS. Our solution follows the same principals and are constructed in the same manner. It will not be necessary to change the architecture, just incorporate the suggested security mechanisms in already established modules and add some new ones. The solution is modular and can be used stepwise, adding more modules if you need them, as long as the baseline is present. This dynamic property is useful since MIDAS is to be used in a wide range of applications and new demands can arise in unforeseen domain areas.

If we compare the solution to other proposed MANET solutions, our solution makes it easier to extend the network since the web-of-trust allows new nodes to be included in the network at any time. It is also possible to take advantage of preparations made beforehand, like for instance distribute certificates before network setup. This dynamic network benefit is also in accordance with MIDAS stated goals, which aims for a fully distributed network where people can join or leave whenever.

8.8.2 Pitfalls

The suggested security solution has been composed with certain assumptions about the middleware and is designed based on our understanding of the MIDAS documentation. The MIDAS documentation was not completed at the time of reading so changes can occur. This is a possible pitfall if the solution is used without verification from the middleware designers, programmers and security specialists.

It is still possible to harvest usage information by participating in the network, downloading all data available. It is also impossible to be anonymous due to the public keys which are distributed by the key server. More work needs to be done to identify vulnerable information and apply the appropriate security measures to protect them. The identification of vulnerable information will probably involve making a policy stating how information should be treated and analysis of the different domain models. Which security measures that is needed can then be researched.

Additionally it is necessary to acknowledge that our solution does not fulfill all the requirements, and the requirements might change over time. We have given a solution that we think will improve the overall security while preserving the usability.

8.8.3 Further issues that need to be resolved

When the solution is implemented into MIDAS some issues needs attention. In addition to verification of the implemented security, as pointed out above, a review of the new system with security included should be performed by a security analyst. The review should look for the weakest link in the resulting system and do a risk analysis using an updated threat model. This should of course be a part of the development life-cycle to build security in, rather than adding it later. The importance of early involvement and building security in is demonstrated well by Gary McGraw in his book on the subject [30].

Another major issue that our solution we didn't take care of in a satisfactory way is the routing protocol and protection of the network layer. The suggestion of signing packets at the network layer is viewed as too problematic due to the overhead it creates. The routing protocol is also an issue as it distributes every address for every node to all other nodes and is not secured. This makes the routing protocol the next priority, if our solution is implemented and security requirements are high. Routing and communications technology is not our area of expertise which is why we have not emphasised it in this thesis.

8.9 Coverage of functional security requirements

Not all security requirements from Chapter 5 are fulfilled by our proposed solution. The functional security requirements (Requirement SR 1 through SR 26) are discussed in this section, and the ethical issues (Requirement SR 27 through SR 35) are discussed separately in Section 8.10.

Some of the requirements are difficult to solve with security mechanisms, because they involve usability and user interface specifics. Examples of this are requirements SR 6

through SR 9 examples of this, and they have not been addressed.

Table 8.1 shows which configurations solve which requirements. Notice how baseline is a prerequisite for the other configurations, so a requirement solved by baseline is automatically solved in the others. We see that requirements SR 2 and SR 10 can be implemented by developers on an application level with the baseline configuration in place, but this does not mean that baseline solves them, which is why they are not checked in the table.

There are three functional security requirements that our solution can't solve (Requirements SR 6 through SR 9 are not viewed as functional requirements). Requirement SR 17 is a requirement which stems from a concern in the MIDAS documentation. It is however, the equivalent of having a back door in the security and these types of requirements can be dangerous to implement.

The other requirements are nicely spread out on the different configurations which were expected as they cover different types of security considerations. If we include the library functions we have solved 19 out of 22 functional requirements, three functional requirements are not solved and four requirements are not possible to solve with security building blocks.

8.10 Ethical security requirements

In Chapter 4 we presented ethical issues. From those issues came requirements which were taken into account when the security solution was created. In this section we will discuss how ethical issues, through the requirements, have been taken care of. We will also discuss why some ethical issues have been difficult to address.

8.10.1 Coverage of ethical issues

The requirements that have been covered well are the following:

SR 29 This requirement is covered if signing of database operations as presented in Section 7.3 is used. As described this will protect the integrity of information contained in MDS at row level.

SR 34 We have not specifically written anything about this requirement, but this can be achieved by an user policy agreement on startup of the middleware. For instance when new applications are deployed, the middleware can warn the user of dangers when contributing potentially sensitive information and show terms of agreement which must be accepted.

We see that the percentage of requirements that are covered is low, which as stated earlier is due to the wording, or the difficult MANET environment. The requirements that are covered are the ones that are concerned with integrity and information awareness. Integrity is the security primitive we have put in most effort to achieve.

	Baseline	Pre-installed certificates	Singed database operations	Signed context and permission certificates	Library functions
SR 1	X	X	X	X	
SR 2					
SR 3	X	X	X	X	
SR 4	X	X	X	X	
SR 5					
SR 6					
SR 7					
SR 8					
SR 9					
SR 10					X
SR 11			X	X	
SR 12			X	X	
SR 13				X	
SR 14				X	
SR 15			X	X	
SR 16				X	
SR 17					
SR 18	X	X	X	X	X
SR 19					X
SR 20					X
SR 21					X
SR 22	X	X	X	X	X
SR 23	X	X	X	X	X
SR 24	X	X	X	X	X
SR 25					X
SR 26					X

Table 8.1: Which configuration solves which security requirement

8.10.2 Ethical issues not taken care of

SR 27 Sensitive information can be protected, but the middleware is not responsible for protecting it. Application developers can use the library functions for encryption of database rows on data they think is sensitive. It is not possible for a user to protect specific data beyond that.

SR 28 This requirement is partially connected to the previous requirement. An access control scheme exists and is put in the configuration described in Section 7.4, but the protection of sensitive data still depends upon the application developers and incorporation of row level encryption.

SR 30 This requirement states that context information can be sensitive and a protection mechanism should therefore be available. This requirement is not met by any of the suggested security solutions, mainly because it would render CAM useless if we encrypted the data and the aggregated context information is distributed with the routing protocol packages.

SR 31 We have not given a routing protocol solution which anonymises users and the message encryption described in 7.5 is not default enabled. In order to fulfill the requirement the routing protocol must be protected and application developers must use secure messaging.

SR 35 There aren't many default settings to speak of, but those who are available are not a part of the baseline configuration, which makes it difficult to argue that this requirement is fulfilled. This requirement is not only about the middleware, but also a note to all application developers.

8.11 Suggested configurations for MIDAS PoC scenarios

The scenarios described in Section 3.2 were chosen by the MIDAS project team because of their different requirements. We therefore use them as examples for which configurations we think are appropriate to use in each scenario. Likely attackers from Section 4.4 are also mentioned.

As long as the *Sports scenario* only use applications for information sharing amongst the spectators and from the domain responsible it will suffice to use the proposed Baseline configuration. With it comes the possibility of auditing. If other applications, written for the contestants, need to protect sensitive data it will not have the large user mass that the other applications for spectators have, so application level security should be used. This is mainly because a more secure configuration would impose possible performance issues for the spectator applications. The typical attacker in the *Sports scenario* is the *script kiddie*, since there is little economic gain of attacking the applications, nor sensitive information in the system. This means that determined attackers have little to gain from attacking applications in the *Sports scenario*.

The *Emergency scenario* will contain sensitive information and can be brought about from a deliberate evil act, meaning someone has the incentive to do harm. Possible attackers here are external determined attackers who will attack the system to disrupt

or corrupt information. It is also possible to envision applications where victims try to manipulate sensor information to get rescued first. Both types of attackers will try to alter data and try to get information they aren't supposed to get. We suggest that this scenario uses the full extent of our configurations and use library functions as well to harden the applications further. Even with our suggested security the applications in this scenario will be vulnerable, due to the unresolved issues presented in Section 8.8.3.

8.12 Division of responsibility between application and middleware

We have designed the MIDAS middleware security so that it supports a library of security functionality, it does not dictate what mechanisms applications should use. This is because different service domains have varying security needs. It's up to application developers to utilise the needed security functionality. Some security mechanisms should be taken care of by the middleware because they are useful for a lot of application scenarios. Others are less frequently needed and could be implemented at the application level instead of in the middleware.

The argument for keeping some functionality out of the middleware is the principle of keeping it simple. A middleware API should be easy to use and understand, and this is more important for security functionality since wrong assumptions about what the functionality really does might lead to insecure applications.

Message received verification is an example of a security mechanism that might not be needed in most application domains. It is also a lot of things to consider, like how to manage lost messages, timeouts and refusing to answer, making it difficult to implement and unnecessary for general purpose applications. We think that this service is best to implement in applications that need it on top of the MIDAS middleware. Our simple form of verification (described in Section 6.5.5) sends out a message with a onetime nonce and expects the receiver to reply with a signed version of this nonce when the messages is read. Since MIDAS does not guarantee delivery the sender must have a timeout after which it will assume the message was lost. Whether or not to resend must be up to the application developer or the user.

Other mechanisms should be enforced by the middleware. An example would be the row level signatures. When performing operations on a table which is defined in MDS to require a signature, verification and signature generation should be performed by the middleware without involvement from application developers. Primitives that protect the routing protocol should also be enforced by the middleware.

If our solution is implemented in MIDAS a thorough evaluation should be performed, which examines and chooses which mechanisms to support and which to drop.

8.13 Choice of technology

In Section 6.1 we proposed to use PGP to enable the web-of-trust model. The obvious alternative was to use X.509 which is more common in commercial PKI systems. There are several advantages with using PGP. The normal operation of PGP is the web-of-trust model which is necessary for the decentralized model in MIDAS. PGP also has a public keyring in a defined format which holds the trust relations of a single user, and can be distributed freely. This makes implementing a central key server in MDS straightforward. Another attractive feature is that PGP provides efficient end-to-end message encryption for multiple recipients, by generating a onetime symmetric session key and only encrypting the session key using public key cryptography. This is faster than encrypting the entire message multiple times using only public key cryptography (more data to encrypt and public key cryptography is significantly slower than symmetric keys).

X.509 certificates could still be used if this is more desirable due to other implementation constraints. An custom extension to X.509 makes it possible to realize the web-of-trust model with a so called *cross-certified mesh* described by Linn [27] [15]. Here participants in a network can sign each others X.509 keys to generate trust relations. A key server capable of distributing public certificates and trust relations will then have to be implemented in the middleware. A scheme for encrypting messages for multiple recipients will also have to be made. The upside is that X.509 certificates are supported directly by Java and has widespread use in commercial PKI systems.

A third option is to implement the needed cryptographic libraries manually. Neither PGP nor X.509 is required for our solution to work. As long as the web-of-trust model is covered and nodes can be uniquely identified, our solution will be possible to implement. This solution should only be considered if using PGP or X.509 proves to have to much performance impact on the overall system. Writing custom cryptographic systems (and especially if low key lengths have to be used to keep performance up) are difficult at best and need to be done by professionals.

8.14 Web-of-trust model

One inherent problem of the web-of-trust model is that different nodes will evaluate access permissions differently based on their individual trust networks (which contradicts Requirement SR 15). This problem does not exist using X.509 certificates without extensions because of the strict hierarchical structure.

The problem occurs when one node set access restrictions on some data (e.g. access for *Police officer*) and leaves it to another node to enforce these restrictions (e.g. a storage node). The storage node can only verify permissions based on its own public keyring and trust network.

The novel approach to solve this would be to let the original node handle all access requests. This will create a choke point and limit availability in partitioned networks. If a trusted node (or set of nodes) are present in the network, then the responsibility can be given to these. This however conflicts with the requirement of decentralized control. More work should be done looking at existing research and new ways to

solve this.

Another problem with the web-of-trust model arise when the number of signed keys are sparse. Finding certificate chains to verify nodes in this network become impossible. This would easily be the case in the sports scenario where most of the spectators will have little or no trust in the network. In the sports scenario this is of little relevance since spectators mostly need access to public information which isn't protected. In other scenarios this might become an issue.

One solution to this problem is to make key signing as easy as possible while still maintaining security, so that the trust network can be expanded quickly. Nodes in reach of each other (in the same local area network) could automatically exchange keys so that signing becomes only a matter of verifying fingerprint and identity.

Chapter 9

Conclusion and Future work

9.1 Conclusion

According to the problem statement given in the introduction to this thesis the objectives was to: Identify security needs in the MIDAS middleware, and evaluate possible solutions.

We identified the security needs by studying the MIDAS documentation and papers written about security in other similar systems. The case study approach gave us a deeper understanding of MIDAS and enabled us to use prior knowledge to identify the possible weaknesses and threats against MIDAS. The weaknesses and threats along with ethical issues resulted in a list of requirements that answers the first part of the problem statement.

The possible solutions we have given consist of security building blocks that can be combined in different configurations which represents different levels of security. The solutions were then discussed to point out what problems they address. The number of covered requirements were: 19 of 22 functional requirements and 2 of 7 ethical requirements.

Chapters 6, 7 and 8 presents security building blocks, different configurations and the discussion answers the second part of the problem statement. The evaluation of possible solutions end with a feasibility discussion where we state that the storage overhead is acceptable, but cryptographic operations will increase the network latency which will need closer investigation and is a trade-off situation between usability and security.

We found that our proposed solution requires only minor architectural changes, and must be integrated into existing components. It is possible that an architectural change will be necessary to protect the routing protocol, but we cannot yet conclude anything in this direction.

The division of responsibility, what is done by whom, between the MIDAS middleware and applications is not entirely clear. Our suggestion is to make the digital signing a middleware responsibility, where a developer can specify that some data needs integrity and the middleware automatically signs and verifies data. Other security parts are available as library functions for the application developers, but they are not mandatory. This is equivalent to a shared responsibility between the

middleware and the applications as the functionality is available, but when and how to use it is up to the application developers.

9.2 Future work

When MIDAS goes past the research stage and is ready for commercial use it will have to make sure security is managed, and clear guidelines for what is acceptable and what isn't have to be worked out by domain responsibles. Future work in the perspective of MIDAS and security analysis requires at least that security expertise is included into the design and development team to make sure it is done right.

When application developers develop applications it would be advisable to do a risk analysis for the task and domain at hand, which cannot be done in general terms, for all domains at once. The results of a risk analysis can then be used to select the proper security mechanisms to mitigate the risks. Our thesis can help both to identify typical threats and to select security mechanisms.

Future work that should be performed researching security in MIDAS include:

Routing protocol The routing protocol should be researched further to perhaps find a way to secure it, or if this proves to be impossible, it should be redesigned. This is a big task that should be done by a network and security specialist.

Detection of misbehaving nodes We described in Chapter 6 a way of detecting misbehaving nodes. Whether or not this detection method is effective or not needs more research by simulation and testing. This research should reveal if the method catches misbehaving nodes and if there is a problem with false positives.

Selection of algorithms Our thesis gives the security design and shows how it is possible to integrate security, but remains algorithm neutral. We have shown that solutions exist which can implement the building blocks, but when the implementation starts specific algorithms must be selected, and a choice between developing software or using existing off the shelf components must be made based on a deeper investigation of the market.

Ontology based trust Ontology based trust enables trust to be limited to specific terms defined in the ontology. By using ontology based trust you can trust someone for a given period of time, in those areas where he is trustworthy, instead of trusting him unconditionally. This form of trust management should be researched as it can add degrees of trust and propagation of trust in the network [21].This form of trust adds a new level of trust relationships on top of the web-of-trust we have presented in this thesis.

Different viewpoints in the web-of-trust Section 8.14 discussed the viewpoints of different nodes and how they would evaluate access permissions. How to solve the viewpoint problem without creating choke points and being dependent upon central nodes needs more work.

Proof of concept implementation A proof of concept implementation of the baseline configuration should be done. The implementation will prove if it is feasible to integrate security into the MIDAS middleware the way we have suggested.

Performance evaluation A performance evaluation of encryption and signing algorithms should be done. The performance evaluation can be done at two levels; separate from MIDAS by testing the algorithms on selected devices, and integrated in the proof of concept. The first type is easier and will give a hint as to whether it is an acceptable overhead or not, but the second type will yield a real life situation where unexpected side-effects can be identified as well.

Bibliography

[1] Key management for encrypted data storage in distributed systems. In *SISW '03: Proceedings of the Second IEEE International Security in Storage Workshop*, page 20, Washington, DC, USA, 2003. IEEE Computer Society.

[2] Ralph Gross Alessandro Acquisti. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *Privacy Enhancing Technologies*, pages 36–58, springerlink.com, 2006. Springer Berlin/Heidelberg.

[3] Louise Barkhuus. Privacy in location-based services, concern vs. coolness. 2004.

[4] Levente Buttyan. Mobility helps peer-to-peer security. *IEEE Transactions on Mobile Computing*, 5(1):43–51, 2006. Member-Srdjan Čapkun and Senior Member-Jean-Pierre Hubaux.

[5] Stefano Campadello. Peer-to-peer security in mobile devices: A user perspective. *p2p*, 00:252–257, 2004.

[6] Licia Capra. Engineering human trust in mobile system collaborations. *SIGSOFT Softw. Eng. Notes*, 29(6):107–116, 2004.

[7] Germano Caronni. Walking the web of trust. In *WETICE '00: Proceedings of the 9th IEEE International Workshops on Enabling Technologies*, pages 153–158, Washington, DC, USA, 2000. IEEE Computer Society.

[8] Hee C. Choi, Sebastian R. Kruk, Slawomir Grzonkowski, Stankiewicz Katarzyna, Brian Davis, and John G. Breslin. Trust models for community-aware identity management. 2006.

[9] Tom Coffey and Puneet Saidha. Non-repudiation with mandatory proof of receipt. *SIGCOMM Comput. Commun. Rev.*, 26(1):6–17, 1996.

[10] Landon P. Cox and Brian D. Noble. Samsara: honor among thieves in peer-to-peer storage. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 120–132, New York, NY, USA, 2003. ACM.

[11] Anwitaman Datta, Manfred Hauswirth, and Karl Aberer. Beyond "web of trust": Enabling p2p e-commerce. *cec*, 0:303, 2003.

[12] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340, sep 1989.

[13] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity, 2002.

[14] Archie D. Andrews Dr. Jack Stinson, Stephen V. Pellissar. Trust model - defining and applying generic trust relationship in a networked computing environment. May 2000.

[15] RFC 4158: Cooper et al. Internet x.509 public key infrastructure: Certification path building, 2005.

[16] RFC 1847: J. Galvin, S. Murphy, S. Crocker, and N. Freed. Security multiparts for mime: Multipart/signed and multipart/encrypted, 1995.

[17] Git user's manual: The object database. http://www.kernel.org/pub/software/scm/git/docs/user-manual.html#the-object-database. Accessed 2008, May 22th.

[18] Gnupg made easy (gpgme) - gnupg.org. http://www.gnupg.org/gpgme.html. Accessed 2008, May 29th.

[19] RFC 2634: P. Hoffman. Enhanced security services for s/mime, 1999.

[20] RFC 2459: R. Housley, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure certificate and crl profile, 1999.

[21] Jingwei Huang and Mark S. Fox. An ontology of trust: formal semantics and transitivity. In *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, pages 259–270, New York, NY, USA, 2006. ACM.

[22] Jean-Pierre Hubaux, Levente Buttyán, and Srdan Capkun. The quest for security in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 146–155, New York, NY, USA, 2001. ACM.

[23] Cem Kaner, James Bach, and Bret Pettichord. *Lessons Learned in Software Testing*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[24] RFC 1422: S. Kent. Privacy enhancement for internet electronic mail: Part ii, 1993.

[25] Ashraf Khalil and Kay Connelly. Context-aware telephony: privacy preferences and sharing patterns. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 469–478, New York, NY, USA, 2006. ACM.

[26] RFC 1510: J. Kohl and C. Neuman. The kerberos network authentication service (v5), 1993.

[27] John Linn. Trust models and management in public-key infrastructures. November 2000.

[28] Joseph Reagle Lorrie Faith Cranor and Mark S. Ackerman. *The Internet Upheavel: Raising Questions, Seeking Answers in Communications Policy*. The MIT Press, Cambridge, Massachussetts London, England, 2000.

[29] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 255–265, New York, NY, USA, 2000. ACM.

[30] Gary McGraw. *Software Security: Building Security in*. Addison-Wesley, 2006.

[31] Middleware platform for developing and deploying advanced mobile services. http://www.ist-midas.org/documents/project_summary_midasv11.pdf. Accessed 2008, May 6th.

[32] Ethan Miller, Darrell Long, William Freeman, and Benjamin Reed. Strong security for network-attached storage. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, page 1, Berkeley, CA, USA, 2002. USENIX Association.

[33] George R. Milne. Privacy and ethical issues in database/interactive marketing and public policy: A research framework and overview of the special issue. *Journal of Public Policy & Marketing*, 19:1–6, Spring 2000.

[34] Dictionary for oma specifications v2.1 approved version 2.4 – 25 july 2006. http://www.openmobilealliance.org/document/OMA-ORG-Dictionary-V2_6-20070614-A.pdf. Accessed 2008, April 16th.

[35] Open mobile alliance overview: Fostering worldwide growth in the mobile services market. http://old.openmobilealliance.org/docs/OMA-Public%20Overview-2004-03.pdf. Accessed 2008, April 16th.

[36] Thomas Plagemann. Design: Middleware for connectivity and information sharing, design of the midas data space. D2.2 Part II, January 2007.

[37] Matija Puzar, Jon Andersson, Thomas Plagemann, and Yves Roudier. Skimpy: A simple key management protocol for manets in emergency and rescue operations. In Refik Molva, Gene Tsudik, and Dirk Westhoff, editors, *ESAS*, volume 3813 of *Lecture Notes in Computer Science*, pages 14–26. Springer, 2005.

[38] Matija Puzar, Thomas Plagemann, and Yves Roudier. Security and privacy issues in middleware for emergency and rescue applications. *MODIES 2008, Workshop on MObile and DIstributed approaches in Emergency Scenarios*, January 2008.

[39] Santiago Pérez. First prototype: Middleware for connectivity and information sharing. D2.2 Part I, February 2008.

[40] Nishkam Ravi, Chandra Narayanaswami, Mandayam Raghunath, and Marcel-Catalin Rosu. Securing pocket hard drives. *IEEE Pervasive Computing*, 6(4):18–23, 2007.

[41] K.A. Remley, C.A. Grosvenor, R.T. Johnk, D.R. Novotny, P.D. Hale, M.D. McKinley, A. Karygiannis, and E. Antonakakis. Electromagnetic signatures of wlan cards and network security. *Signal Processing and Information Technology, 2005. Proceedings of the Fifth IEEE International Symposium on*, pages 484–488, 18-21 Dec. 2005.

[42] Simon Rieche, Klaus Wehrle, Olaf Landsiedel, Stefan Gotz, and Leo Petrak. Reliability of data in structured peer-to-peer systems. In *HOT-P2P '04: Proceedings of the 2004 International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 108–113, Washington, DC, USA, 2004. IEEE Computer Society.

[43] Securing data at rest - developing a database encryption strategy. Technical report, RSA Security Inc., 2002.

[44] Subir Kumar Sarkar, T G Basavaraju, and C Puttamadappa. *Ad Hoc Mobile Wireless Networks: Principles, Protocols, and Applications*. Auerbach Publications, 2007.

[45] Robert R. Schaller. Moore's law: past, present, and future. *IEEE Spectr.*, 34(6):52–59, 1997.

[46] Albrecht Schmidt, Tom Gross, and Mark Billinghurst. Introduction to special issue on context-aware computing in cscw. *Comput. Supported Coop. Work*, 13(3-4):221–222, 2004.

[47] Bruce Schneier. *Beyond Fear: Thinking Sensibly about Security in an Uncertain World*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[48] Ludwig Seitz, Jean-Marc Pierson, and Lionel Brunie. Semantic access control for medical applications in grid environments. In *Euro-Par*, pages 374–383, 2003.

[49] William Stallings. The pgp web of trust. *BYTE*, 1995.

[50] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4):149–160, 2001.

[51] Mika Ståhlberg. Radio jamming attacks against two popular mobile networks.

[52] Ingrid Svagård. Overall architecture of the midas middleware: Midas requirements and testing. D1.2 Part IV, Februar 2008.

[53] Ingrid Svagård. Overall architecture of the midas middleware: Other design concerns. D1.2 Part III, Februar 2008.

[54] Ingrid Svagård. Overall architecture of the midas middleware: Overall problem definition and approach. D1.2 Part I, Februar 2008.

[55] C.K. K Toh. *Ad Hoc Wireless Networks: Protocols and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[56] Jeremy Wagstaff. *Loose Wire: A Personal Guide to Making Technology Work For You*. Equinox Press, 2006.

[57] Roy Want. You're not paranoid; they really are watching you! *Pervasive Computing, IEEE*, 6(4):2–4, Oct.-Dec. 2007.

[58] Ford Long Wong and Frank Stajano. Multichannel security protocols. *IEEE Pervasive Computing*, 6(4):31–39, 2007.

[59] Dongyan Xu, Mohamed Hefeeda, Susanne Hambrusch, and Bharat Bhargava. On peer-to-peer media streaming. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 363, Washington, DC, USA, 2002. IEEE Computer Society.

[60] S. Yi and R. Kravets. Moca: Mobile certificate authority for wireless ad hoc networks. *The 2nd Annual PKI Research Workshop (PKI 03).*

[61] Philip R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995.

Glossary

AC Access Control

API Application Programming Interface

CA Certificate authority

CAM Context addressable messages

CBR Context based routing

CFG Common Functions

CKB Context knowledge base

CPU Central Processor Unit

CRT Communication and routing

CTX Context engine

CXS Context Space

DAC Discretionary access control

DHT Distributed Hash Tables

DoS Denial of Service

DPL Deployment

GPGME GnuPG Made Easy

JNI Java Native Interface

MAC Mandatory access control

MANET Mobile ad-hoc network

MDS MIDAS Data Space

OMA Open Mobile Alliance

P2P Peer-to-peer

PDA Personal Digital Assistant

PGP Pretty Good Privacy

PKI Public Key Infrastructure

PoC Proof of Concept

RBAC Role based access control

RF Radio Frequency

SOA Service Oriented Architecture

TTL time to live

A Security challenges from the MIDAS documentation

C1	Information may be sensitive and not intended for all users of the network. Mechanisms are needed to protect sensitive information when stored and when communicated, and there must be limitations as to who gets access to what information.
C2	Since emergency and rescue operations may take place anywhere one should not rely on central units outside the emergency area to provide for instance authentication of users.
C3	It is difficult to know beforehand who will need access to what information. It is not possible to know where the emergency will take place and who will be involved.
C4	Roles may also be changed along the way, e.g. as more rescuers come to the scene.
C5	Especially if the emergency was intentional (e.g. terrorist attack) it is important to limit possibilities for malicious users and outsiders to disrupt services or spread incorrect information and messages.
C6	Assuring privacy will in many cases be less important than to perform effective emergency and rescue operations. This means that in some cases it may be necessary to have the possibility to bypass access control restrictions, something that will result in weaker overall protection of information.
C7	Mechanisms that assure non-repudiation of orders given may be needed.
C8	Receive receipts that information has been received/read may be needed.
C9	The security solutions offered by MIDAS must be easy to set up, to allow for rapid setup and configuration.
C10	The number of users and the limited possibilities to establish access credentials beforehand poses challenges for authentication and access control.
C11	The authentication solutions must be simple, meaning that they are fast and easy to use, while assuring that users can get access to services in a controlled manner.
C12	Some services offered may use context information on the users (e.g. location of users to get information on when their favourite biker are passing). Users may want to protect this information to ensure privacy.
C13	Users may want assurance that information they send or information on which services they use are not available to other users, to ensure privacy.

C14	If users pay for services, it is important to assure that users cannot simply share their access with e.g. family and friends.
C15	If users pay for services, information security will be a major concern when implementing payment mechanisms.
C16	Ad-hoc networks also come with additional challenges since attackers can become a router, and thereby attack the routing itself by announcing false routing packets.
C17	It is also possible to attack the data pane by dropping, replaying or injecting packets.
C18	For routing of sensitive information [...] there is also a need to make sure that e.g. attackers or malicious users cannot get access to large amounts of information by placing themselves in central positions in the network. Trust between nodes becomes a main issue, and authentication of nodes, not only users, becomes important.
C19	When designing and implementing security solutions, it must be taken into account that terminals can be stolen or lost.
C20	With such devices [laptops, mobile phones and PDAs] users have physical control of the component, meaning that it should be measures in place to assure that users cannot e.g. change their access rights by changing files or similar in their terminal.
C21	By performing general attacks on nodes that are part of the MIDAS network attackers may e.g. get access to information and services.
C22	Access control to information must not only be present at e.g. Super Nodes, but must take place all places information and services are available.

Table A-1: Security challenges from the MIDAS documentation

B Security requirements

	Demand	Comment
SR 1(M)	Users must be able to report lost or stolen devices at which point they would be considered untrusted by the network. Reports should be impossible to deny or withdraw. Special care to prevent denial of services has to be taken.	
SR 2(M)	Data stored on nodes should not be easily available if device is stolen or lost.	C19
SR 3(M)	Users should not be able to elevate their permissions by changing files locally on device. This includes modifying the middleware software itself.	C20
SR 4(M)	The devices should verify the identity of the user (user authentication) so that stolen devices are rendered useless.	C20
SR 5(M)	General remote attacks on nodes should not reveal sensitive information to an attacker.	C21
SR 6(H)	Security solutions in MIDAS should be easy to set up, both to allow rapid network deployment and eliminate the need for technical personnel.	C9
SR 7(H)	Authentication mechanisms must be simple to use for non-technical users.	C11
SR 8(M)	Tools for managing permissions must be intuitive and self explanatory	.
SR 9(M)	Most security settings should be managed with little or no user involvement.	
SR 10(H)	Sensitive information must be protected when stored and communicated.	C1
SR 11(H)	There must be limitations as to who gets access to what information.	C1

	Demand	Comment
SR 12(H)	Access granting should be possible both off-line and on-site.	C3
SR 13(M)	The middleware must support well defined roles.	C4
SR 14(L)	Access should be granted to a given role or user. Roles might change dynamically as users leave or arrive.	C4
SR 15(M)	Every node should be able to evaluate access permissions and the result the of evaluation must be the same on all nodes. This includes authentication, role and permission verification.	C2, C22. Evaluation changes dependent on who you trust, so this might be difficult to achieve in practice.
SR 16(M)	Users (or applications) providing information into the network should be able to set access restrictions on the data they provide.	
SR 17(M)	Access restrictions should in some cases be possible to bypass.	C6 This applies in the emergency rescue scenario when lives are at risk.
SR 18(H)	Nodes must be able to participate in the network without access credentials having to be established beforehand.	C10
SR 19(H)	Untrusted nodes should be able to handle (store, route) sensitive information.	That is to say they need to be able to receive messages they do not have access to read.
SR 20(M)	MIDAS should protect packets from being modified in transit. Malicious nodes should not be able to disrupt service, gather information, send false messages or corrupt the routing protocol (includes dropping, replaying or injecting packets).	C5, C16, C17
SR 21(L)	Malicious nodes trying to disrupt the service should be detected and locked out. This has to be done in a way so that obvious denial-of-service attacks are not possible.	

	Demand	Comment
SR 22(L)	Malicious nodes should not be able to gather sensitive information when routing messages.	C18
SR 23(H)	MIDAS must be able to unambiguously determine the source of a message.	node and maybe also user.
SR 24(M)	The trust model between peers in the MIDAS network must support ad-hoc setup and match any hierarchical structure.	Pure tree-like structures with a single root node might be to restrictive
SR 25(L)	Some messages should force returning a receipt upon receiving/reading.	C8
SR 26(M)	MIDAS needs to support secure communication between nodes trusting each other without having to trust every routing node in between. This includes message integrity, data protection, non-repudiation and protection against replay attacks.	C7
SR 27(H)	Sensitive information (as seen from the system) must be secured to prevent unwanted exposure.	
SR 28(H)	The middleware must protect sensitive information from unauthorized access.	
SR 29(H)	The middleware must protect the integrity of information.	
SR 30(M)	Some context information is considered sensitive and should therefore be protected. This includes aggregated data.	C12
SR 31(M)	Usage of services and peer interaction is sensitive information and must therefore be protected.	C13, This requirement is based on privacy issues when for instance a contestant in the sports scenario requests medical attention. This should not be broadcast to opponents team leaders.

	Demand	Comment
SR 32(H)	Personal information should be deleted at the request of data owner.	
SR 33(M)	Users have to be able to see what information they share to others.	
SR 34(H)	Users should be warned about what the information they give will be used for.	
SR 35(M)	Default settings must favour a high degree of privacy.	

Table B-2: Security requirements specification

C Example on signed context certificate

Example of what an xml-based context certificate might look like. The certificate have been signed with PGP using cleartext-sign. This yields a human readable output, but uses more space. A binary format signature is also possible. Sizes of both types of files are given below.

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

<context-certificate>
    <unique-identifier type="pgp-fingerprint">
        AD82 54B3 39CE 673A 1406  FD07 5CA4 50DE 52B7 487E
    </unique-identifier>
    <context>
        Police officer
    </context>
    <valid-to dateformat="utc">
        Thu Jun 29 09:59:04 UTC 2008
    </valid-to>
    <signing-authority type="pgp-fingerprint">
        73DA 818A B0DF BCB2 187B  80B6 76CC F0C1 76BD 4309
    </signing-authority>
</context-certificate>
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iD4DBQFIPn9edszwwXa9QwkRAhV2AJ93MLkPwEw26RR2yTbpxybv98CI6wCYuc3H
Q0Skd/hYBPWltkxds7t37w==
=pUjH
-----END PGP SIGNATURE-----
```

Total size of xml input: 444 bytes

Total size of signed input: 680 bytes

Total size of signed certificate using binary format: 377 bytes