Access Control in Multi-Party Wireless Sensor Networks

Jef Maerien, Sam Michiels, Christophe Huygens,
Danny Hughes, and Wouter Joosen

iMinds-DistriNet, KU Leuven, 3001 Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Abstract. Emerging real world WSNs seldom exist as single owner, single application, isolated networks, but instead comprise of sensor nodes owned by multiple parties. These sensors offer multiple services to users locally or across the Internet, and travel between multiple WSNs. However, users should only have access to a limited subset of services. Due to a need for direct interactions of users with nodes, authentication and authorisation at the node level is critical. This paper presents an access control infrastructure consisting of three parts: 1) an authentication protocol to ensure authenticity of messages, 2) a role based authorisation framework to perform access control, and 3) a user management service to enable user and permission management. A prototype implementation on the ContikiOS demonstrates the validity and feasibility of node local role based access control on low power micro-controllers.

Keywords: Security Middleware, Wireless Sensor Networks, Authentication, Authorisation, Role Based Access Control

1 Introduction

Many WSN use cases have been proposed in a wide array of application domains such as agriculture [1], logistics [2] and domotics [3]. While most first generation research focused on single application, single owner networks, it is clear that future sensing infrastructures consist of platforms owned by different parties. These nodes run multiple concurrent services[4, 5] and can travel between multiple wireless networks, owned by other parties. These third parties provide Internet connectivity, giving users both local and remote access to node services. These users also want to deploy new configurations and applications onto these nodes together with party specific access policies [6].

This paper defines a node service as a software component which enables a set of related operations to be performed on the node. Users communicate with the service by sending request messages, indicating what service and operation they want to use, and receiving reply messages. Typical services in WSNs include a temperature service, offering information on current and past temperatures, or a lock actuator service, allowing lock opening.

In the described multi-user setting, strong access control is critical in order to ensure the integrity and viability of the system. A minimal access control infrastructure is comprised of three parts: 1) authentication: only known users can access services, 2) authorisation: users only have access to allowed services, and 3) management of users and permissions. Likely each node will have unique access control policies, since most nodes will be used by different parties, each with their own policies regarding application access.

Access control cannot be done at the gateway because, in the local case, the gateway is usually not part of the communication flow. In the remote case, the gateway is not necessarily owned by the same party as the sensor node [6]. The platform owners back-end could perform the necessary access control, although it would introduce significant overhead. The platform owner must then collect all access policies from all parties who currently have some application or configuration installed on the nodes for all their potential users. In some cases it would even be impossible to gather and effectively enforce such policies, for example when no back-end network connection is available.

It is thus required that this access control infrastructure exists at the node level to allow the sensor node to offer its services to users both local in the network, for example customs officer opening a container, or remote across the Internet, for example a logistics provider tracking his container[7]. Current research has investigated access control in WSNs, but often only considers authentication, with limited attention to authorisation and policy management.

The contribution of this paper is an integrated WSN access control infrastructure that allows multiple parties to securely use shared sensor node services running on top of low power micro-controller class devices and manage their access rights in a party specific fashion, which has not yet been tackled in literature. It consists of three parts: (1) an authentication protocol to ensure authenticity and confidentiality of user service request, (2) an authorisation framework using role based access control to ensure only authorised users can access services, and (3) a user management service. An implementation and evaluation of this infrastructure on the Contiki operating system and LooCI middleware demonstrates the validity of the approach.

The remainder of this paper is structured as follows. Section 2 presents a use case in the field of logistics and derives security requirements. Section 3 discusses current related work. Section 4 proposes the access control infrastructure. Section 5 provides details of the implementation and reports on the performance and security evaluation of the system, and finally Section 6 lists some promising avenues for future work and concludes this paper.

2 Use Case

2.1 Logistics Use Case

This section presents a logistics use case to illustrate the requirements as shown in figure 1. In the logistics domain, Logistics Providers (LPs) provide end to

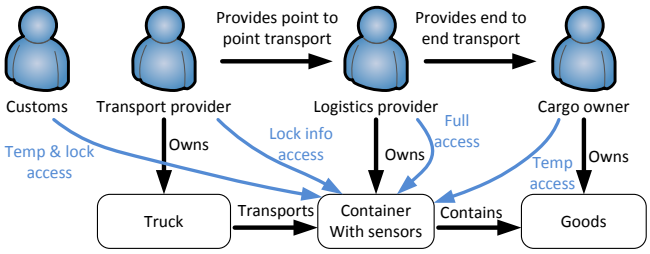

Fig. 1: Overview of the logistics use case.

end transport to cargo owners using containers. In order to do this, the LPs use the trucks or ships of transport providers to carry containers across the world. Such trucks can drive significant distances and cross borders, requiring customs inspection by governments. To provide end to end transport, a logistics provider will use the services of multiple transport providers.

In order to maintain up-to-date information on and control over the conditions of the cargo, the LP equips his containers with nodes that monitor and regulate the conditions inside the container, for example temperature sensors and lock actuators. With this infrastructure, the LP can guarantee that the agreed service level, such as temperature of containers, is maintained by actively and remotely monitoring and regulating container conditions. Other parties closer to the container also want to use the services of these nodes: the transport providers want to ensure that the container remains locked, cargo owners want to validate the transport, and customs officers require that transport happens in compliance with regulations and also need to open the door to inspect the container.

This use case clearly shows that there are multiple parties in the logistics ecosystem, both local in the network, and remotely across the Internet, who want to interact with the sensor node, yet not all of them should have access to the same services. Cargo owners should not open the door, customs officers should not change the cooler settings, and so on.

2.2 Attacker model

This paper considers two types of attackers : Network Attackers (NAs) and Physical Attackers (PAs). NAs are attackers with access to the messages exchanged over the network, in accordance with the Dolev-Yao [8] model. An NA can manipulate, duplicate or create messages, but he cannot break the underlying cryptographic primitives. He aims to gain information on or some kind of control over the network. PAs gain access to the information of the node through physical sensor probing. Standard sensor nodes are incapable of resisting PAs, unless the node is physically secure, by being either encapsulated in tamper proof hardware, or in a physically secure environment.

2.3 Security Requirements

From the preceding use case, this section derives a set of functional and security requirements for the access control infrastructure:

- **Light weight:** sensor nodes are low cost entities with limited processing, energy, and communication resources.
- **Multi-user:** multiple users have to be able to interact with the system.
- **Fine grained access control:** all users should not be able to access all services. Service access depends on the service owner and the service user.
- **Per-node management:** sensor nodes are unique entities and should not be seen as a single homogeneous network of sensor nodes. The lock node and the environment control node should not have the same access policies.
- **Protection against network attacks:** A networked attacker cannot gain access to restricted node services (authorised access only), nor can he modify messages undetected (request integrity).
- **Recovery from physical attacks:** physical attackers can gain access to all security information on the sensor nodes. When this happens, the remainder of the network should be able to recover to normal operation.

3 Related Work

	Symmetric key	public key
Light weight	+	–
Multi-user	-	+
Fine grained access control	-	+
Per node management	-	+
Resistant to node capture	-	+

Table 1: Comparison of related work on authentication in WSNs with regards to requirements.

Many access control systems have been proposed for WSNs. However, most systems only focus on authentication, neglecting authorisation and policy management. Current authentication systems can be divided into two categories: symmetric key and public key systems. Table 1 shows a feature overview. Symmetric key approaches offer light weight authorisation but have issues with multi-user networks and management. Public key approaches provide authentication and sometimes authorisation of user requests, but do so at a very high cost.

The first category is symmetric key authentication. Many algorithms provide authentic broadcast using symmetric keys. Some examples are uTesla [9], TinySec [10], and sAQF [11]. TinySec uses a short Message Authentication Code

(MAC) to authenticate messages using a shared secret key. UTesla uses key chains and delayed key disclosure to ensure authenticity of broadcasts. The sAQF protocol uses key rings in order to perform authentication. With each key of his ring, the user calculates a one bit MAC. Each node has a subset of these keys which it uses to verify message authenticity. These protocols often assume homogeneous networks and do not support multiple users. Furthermore none of these protocols perform any form of authorisation nor policy management and, assuming group distribution of keys, are vulnerable to node capture.

The second category are public key authentication schemes. Messages are authenticated using the private key of the transmitter. Different protocols propose different ways to authenticate the public keys. Most propose a variation of a certificate authority, such as the work of Benenson et al [12]. Ren et al [13] use bloom filters to perform validity checking. He et al [14] propose Priccess, which uses ECC ring signatures to authenticate and authorise users. Yu et al [15] perform access control by encrypting the data with symmetric keys and encrypting these keys with public keys. Only members of a group with the necessary access rights can then decrypt the data using their private keys. This class is better protected against node capture, but does so at significant cost due to asymmetric encryption, This makes them unsuitable for low power micro-controllers. These schemes mention user revocation and key management, and some provide authorisation. However, non of these solutions offer a light weight access control framework allowing fine-grained access policies at the node level.

4 Access Control Infrastructure

This section describes the high level architecture of the access control infrastructure. First this section details the network architecture and the assumptions. Next it looks at the 3 different parts of the access control infrastructure: (1) the authentication protocol, (2) the authorisation framework, and (3) the user management service. Figure 2 shows the components added to the node system: the authentication interceptor, the authorisation layer with a proxy for each service, and the user management service which is comprised of a marshaller, a service execution component and a user database.

4.1 Network Architecture

The system identifies three entities in the network as shown in figure 3: the user platform, the Platform Owners (PO) back-end system, and the node. This paper makes a distinction between users and parties: users are individuals who use node services offered by parties. A party is an administrative entity that groups multiple users together and owns the sensor nodes. Parties offer their nodes' services to their own members and members of other parties. Each user belongs to one party, but can have permission to use the services of other parties.

The user platform is the system from which the user sends his messages to the node platform. This can be a computer, smart phone or sensor node. The

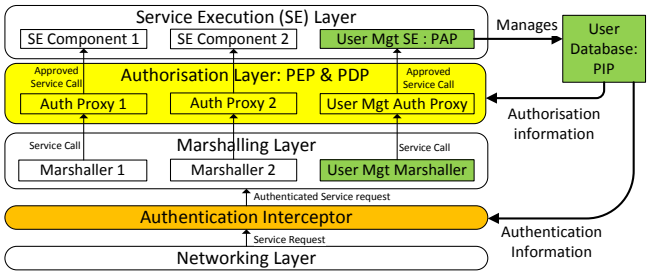

Fig. 2: Node architecture overview with three added parts: authentication interceptor (orange), authorisation framework (yellow), user management service (green)

user must be trusted by the PO. This trust is usually expressed in a contract between the PO and user's party. This contract states which access rights the users of a party receive in exchange for monetary compensation.

The Platform Owner (PO) is the party that owns the node. In the logistics case, this would be the Logistics Provider. The PO offers a back-end system which allows users to request access to specific node. When a user wants to access a node, it securely contacts the PO using Internet security protocols (SSL/HTTPS), using mutual authentication to verify the trust relationship between PO and user. A user can retrieve the PO of a node in several ways: 1) he can access an unprotected node service, which returns the PO's identity, 2) he can know ahead of time whose nodes are present, or 3) he can access a network information service and query which nodes and parties are currently in the network.

When the user knows which nodes he wants to use and who owns them, he sends a request containing the node(s) he wants to use (step 1 in figure 3). This request contains his partyId, and what access rights he wants. If permitted, he'll receive a permission token (step 2). This token is valid for a configurable amount of time ranging from minutes to weeks. When the user wants to start using node services, he must first register himself with the node by sending the token (step 3). Once registered, he can use the node services using his provided userId and key (step 4). The user can use the services until he is removed. The PO must declare in his back-end which users and parties are allowed which permissions in human-readable back-end policies. The node access control infrastructure is agnostic of how these permissions are expressed: they can be expressed in any formal policy language, such as for example XACML [16].

The final part of the network architecture is the node. The node provides certain services which can be accessed by users of the different parties. These services can provide information, reconfiguration or actuation operations. Users of parties which are sufficiently trusted are also able to add new users and manage their access rights by using the user management service. The exact access rights of users are expressed in the contract between party and PO.

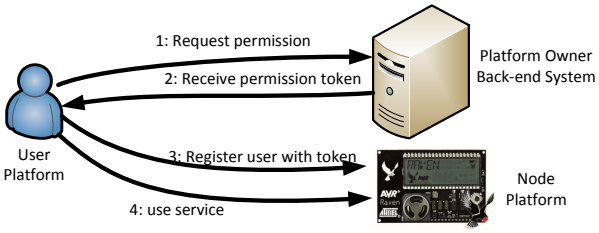

Fig. 3: User registration process.

4.2 Assumptions

The access control infrastructure makes the following assumptions: (1) a secure network layer is present, allowing communication through asynchronous message passing, (2) the node services are separated into a marshaller and service execution component, and (3) each node starts at its owner, where it is securely equipped with the necessary key material and security middleware.

This paper assumes that the WSN uses asynchronous message passing, routed over a secure network layer. This paper is agnostic with regards to the network layer, but requires messages to consist of a payload and headers. The payload contains the functional information, the headers contain non-functional information such as security meta-data.

The protocol requires a separation of any service component into two subcomponents: the marshaller and the Service Execution Component (SEC). The marshaller interprets incoming service requests, accesses the SEC, and serializes the reply. The SEC contains the actual data and logic to perform the requests. This division is common in RPC architectures such as CORBA and RMI.

The last assumption states that the PO can securely install some initial key material onto the node. Without this initial trust setup, no new trust relationship can be deployed. This key material takes the form of a unique symmetric key shared between the PO and the specific node.

4.3 Authentication Protocol

The protocol uses a simple authentication mechanism. Each user is identified by the sensor node using a node specific numeric identifier. The user authenticates and secures his commands using a symmetric encryption algorithm in Counter with Cipher block chaining-MAC (CCM) mode. This allows for the encryption of the message payload with proof of authenticity and integrity of the payload using a Message Authentication Code (MAC). The protocol is agnostic to the actual protocol used. The actual implementation of the protocol uses AES128 due to security, resource requirements and standardisation considerations. The userId and MAC are added to the message as headers. An authenticated timestamp is optionally added to ensure message freshness.

When a node receives a service request, it is intercepted by the Authentication Interceptor (AI) before being delivered to the marshaller. The AI retrieves the userId and MAC from the message headers. It retrieves user information from the user database (Policy Information Point: PIP using XACML terminology [16]). If the userId is known, the AI decrypts and verifies the payload. If validation fails, either because of incorrect MAC or unknown userId, the message is dropped. On success, the request is delivered to the marshaller. When the marshaller sends a reply, it must reattach the userId. The AI intercepts this reply and encrypts and authenticates the payload.

4.4 Authorisation Framework

To authorise a user, an authorisation proxy is inserted between the marshaller, and the Service Execution Component (SEC). This proxy offers the same interface as the SEC, thus can be inserted transparently with minimal effort. The proxy acts as the Policy Enforcement Point (PEP) and Policy Decision Point (PDP). It uses role based access control to authorise a user. The proxy knows the required role that a user must have. This hard-coded role requirement allows for efficient evaluation of access rights, yet is less flexible. The proxy retrieves the user's current roles from the small user database on the node, and verifies whether the user's role is sufficient.

A user can have two types of roles: node roles and party roles. A node role defines the access permission that the user has across the entire node. This allows the PO to compactly declare that a user can view or reconfigure any configuration or service on the node or to perform certain node-wide reconfigurations. A party role defines the role that the user has with that party and is only relevant for applications and configurations owned by that party.

The framework currently distinguishes 5 different access roles, listed in hierarchical order: 1) **no access**: no access to any service, 2) **viewer**: viewing information and configuration, 3) **user**: modifying existing configurations, 4)**manager**: creating and removing configuration, and 5) **administrator**: user management. Currently a higher roles also assumes all access rights of the lower roles. The number of roles and allocation of access rights is a generic framework. Roles and rights can easily be modified to adhere to domain specific requirements.

A user can have one node role, and a role for each party installed on the node. When a user adds an application or configuration, it must be assigned to a party. Either the user's default party is used, or the user specifies on behalf of which party he performs the creation operation. Of course the user needs the necessary service permission in order to create new configurations. Further service request regarding the added application will require that the requesting user has the necessary role to either that party, or node wide.

The authorisation proxy also allows for monitoring the behaviour of users of the system. Monitoring node users allows for 1) detecting potential intrusion attempts, and 2) logging of sensor node usage caused by the different users, allowing chargeback of node usage to the users.

	Logistics Provider user	Transport Provider user	Customs Officer
Lock app : LP	node admin	party viewer	party user
Location app : TP	node admin	party admin	party viewer

Table 2: Overview of the permissions of the different users with the different parties and their applications

An example from logistics as show on table 2: the Logistics Provider (LP) provides the node in the container with a lock application. The LP user has administrator rights to the node and thus this service, allowing him to open, close and manage it. A customs officer has party user permissions to services of the LP. The officer is allowed to use the lock service, namely open and close the lock and view status, but is not allowed to manage which other parties can have access. The Transport Provider(TP) user finally has only party viewer permission. He can only view current lock status.

Suppose the TP installs a localisation component on the node, which queries the truck and broadcasts it to all users registered on the node. The TP user is party administrator, and can administer which parties are allowed to view the location feed, as agreed upon in the contract that the TP and LP need to have signed beforehand. The LP user is a node administrator, so he can also administer who can view the feed on the node, such as cargo owners or customs officers. The customs party is allowed to use services offered by the LP, but only has view rights to applications offered by the TP.

While the current prototype only offers access control based on party roles, a small addition to the system would allow additional attribute based security, enabling parties only rights to certain allowed ranges of applications, even while having node viewer permissions.

4.5 User Management Service

User management is done by the User Management Service (UMS), making it the Policy Administration Point (PAP). The provided methods are: adding and removing users, adding and removing permissions, and updating key material.

The UMS provides two ways to add users, by command or by token. A service method allows users to add new users by command if he has the necessary role. To add a user by token, the user requests PO's back-end to generate a token on his behalf. The user can then send it to the UMS. This token contains following parameters: userId, partyId, node role, party role, key material, timestamp and timeout. The timestamp and timeout of the token ensure that a token will only be valid for a limited amount of time, allowing sensor node recovery. This time is configurable, allowing for tokens with a longer validity. These tokens can be requested ahead of time, and deployed in disconnected networks. The generated token is encrypted with the node's secret key, shared between the PO and the node. The node does not need to contact back-end infrastructure to ensure token

validity. Once a user is added by command or token, the user only needs his userId and key to query node services.

Other provided services include a user revocation service, a role management service, and a key management service. The user revocation service allows users with a node administrator role to remove any user on the node, including themselves. Party administrators are allowed to delete users belonging to their party. The role management service allows users with a party administrator role to add and remove roles of that party from other users. This allows parties to manage access rights of users to their own services, while running on the PO's platform. The key management service allows users to refresh keys after a certain time interval. However this does not ensure forward key secrecy. If a user's key is breached, the attacker can intercept the re-keying message and retrieve the new key. If this is detected, the user has to be removed. The user can be reinstalled by a node administrator or by using a new token.

If a user needs to manage multiple nodes, the same symmetric key can be used to allow group reconfiguration. This reduces the amount of messages the user has to send to perform reconfiguration, but at the cost of security. In case of a node breach, the user's key would be revealed and could then be used to access all other nodes on which the user used the same key.

5 Implementation and Evaluation

The protocol is implemented on top of AVR Ravens [17] running the ContikiOS [18] with IPv6 and the LooCI component middleware. The access control infrastructure can in theory be implemented using other operating systems, such as TinyOS, or other service protocols, such as CoAP. This section details the prototype, performs an evaluation, compares these values to related work and ends with a threat analysis of the proposed infrastructure.

5.1 LooCI Component Middleware

The prototype of the proposed security middleware secures the management of the LooCI component middleware [19]. Both component middleware and access control infrastructure prototype are available at code.google.com/p/looci/. LooCI is a component-based middleware existing out of an execution environment, a component model and an event-based binding model. The LooCI middleware currently supports various platforms including Contiki on AVR Ravens, SunSPOT nodes and the OSGi component model. LooCI consists out of the following parts: 1) The networking layer, 2) the event manager, 3) the component runtime, 4) the management service that is composed of a marshaller, a service execution component and a user database, and 5) the deployment module. The event manager and component runtime function as marshalling layer: the event manager dispatches events to the correct marshalling component depending on event type. The choice of LooCI was made because it supports dynamic application deployment, event based management by multiple users, an event format

with headers and a clean separation between network layer, marshalling layer and service layer allowing for easy integration of access control.

LooCI uses events as the sole mechanism to communicate between components. LooCI events consist of the following parts : (1) sender information, (2) extension headers, (3) event type, and (4) event payload. The event type is a 16 bit value, which identifies what type of content the event carries. Reconfiguration and inspection of the event manager is done by the contacting the management service using management events. Deployment of new applications uses an optimised deployment protocol.

5.2 Implementation details

The prototype intercepts events between the networking layer and the event manager. The userId and MAC are retrieved from the event headers. A 16 bit userId provides a sufficient number of users while ensuring limited overhead. The prototype authenticates and encrypts the events using AES128 in CCM mode with an 8 byte MAC, which provides a sufficient level of security with limited message overhead. The prototype authenticates the sender and receiver IP and timestamp by using the associated data field of CCM.

In order to authorise an event, an addition was made to the LooCI middleware: when an event is dispatched to a marshaller, the dispatching user is recorded by the authorisation middleware. When a marshaller calls any proxy protected service execution component, the authorisation proxy intercepts the call and verifies that the requesting user has sufficient permissions to access the requested service. If the user is authorised, the proxy calls the management SEC. If not, an error code is returned to the reconfiguration manager. When the marshaller sends out a reply, the middleware attaches the userId.

The user management service is implemented as a LooCI component, existing of a marshaller, which interprets received messages, and a SEC, which does the actual user management and contains the user data and access policies. The SEC thus performs both the functions of PIP and PAP in order to reduce implementation size.

5.3 Evaluation of Implementation

A prototype implementation of the protocol was made for ContikiOS [18] on the AVR Raven [17] running LooCI [19]. The AVR Raven is a wireless sensor node with 128kB ROM, 16kB RAM, 20MHz MCU and sleep energy usage of about 1µW. This classifies the device as a low power micro-controller. The tests evaluate the prototype against 6 metrics: message overhead, user registration overhead, message processing time, user installation time, RAM and ROM overhead, and energy cost. These figures are compared to a symmetric key approach (TinySec [10]) and a public key based approach (authenticated querying by Benenson et al. [12]) as shown in table 3. These systems were chosen because they were implemented on a sensor node and listed performance figures.

Comparison criteria	Proposed protocol	TinySec	Auth. querying
Message overhead (bytes)	14	8	20
User reg overhead(bytes)	72	Not applicable	114
Message processing time(ms)	4.64	1.52	Info not available
User installation time(ms)	4.64	Not applicable	440 000
ROM overhead (bytes)	12 147	7 148	45 500
RAM overhead (bytes)	438	728	2 000

Table 3: Comparison of implementation overhead between the proposed access control infrastructure, TinySec and authenticated broadcast

Message overhead: each message is authenticated using AES128-CCM with 8B MAC. The MAC is attached to the message in a secure payload header. The resulting message overhead is comprised of: 1) the userId header: 2B, 2) optional timestamp headers: 4B and 3) the security payload header: 8B. Each header has an additional overhead of 2B. Total: 14B-20B. To compare, TinySec message overhead is 8B, due to smaller MAC (4B). Authenticated broadcast message overhead is 20B.

User registration overhead: a user token consists of the following fields: userId, partyId, nodeRole, partyRole, userKey, timeStamp, timeOut, and MAC. The token has a total message size of 36B. The token is sent as a special user, requiring another 4B. The networking and link layer overhead of sending messages is estimated at another 32B. Total transmitted message: 72B. TinySec does not mention any management of users. The authenticated querying overhead of deploying a user certificate is 114B.

Message processing time: the authentication of a message comprises of a) checking the userId, b) retrieving the user information, and c) decrypting and verifying the authenticity of the message. The most expensive operation is decryption: 4.64ms for a 32B block. All other operations are negligible, taking only several nanoseconds. Authorisation is much faster: only ca 20ns and comprises of a) a proxy interception, b) retrieving user access rights, and c) verifying access rights. In comparison, TinySec evaluated two block ciphers: to encrypt 32B: RC5: 1.04ms, SkipJack: 1.52ms. No data is available on the message processing overhead of Authenticated querying.

User installation time: the installation of a new user using a token requires the decryption of a 32B token, which is again the most expensive operation requiring 4.64ms. TinySec does not provide user or key management. Authenticated querying has a verification overhead of 440s to install a new certificate, mainly due to two ECC operations.

RAM and ROM overhead: table 4 lists the ROM and RAM requirements of the different parts of the implementation. The implementation contains 4 parts: 1) the AES128-CCM encryption algorithm, 2) the authentication interceptor, 3) the authorisation proxy, and 4) the user management service. The total platform has an overhead of 12 147B of ROM and 438B of RAM, which

is comparable to related work: TinySec has an overhead of 7 148B of ROM and 728B of RAM. Authenticated querying has an overhead of 45.5kB of ROM and 2kB of RAM. Note that 39% of ROM and 46% of RAM usage are due to the encryption algorithms, which is shared between all security protocols.

	ROM overhead (bytes)	RAM overhead (bytes)
1 ContikiOS	42 688	9 712
2 LooCI component middleware	24 942	2 644
3 Encryption : AES128-CCM	4 746	200
4 Authentication Interceptor	1 557	129
5 Authorisation proxy	3 296	0
6 User Management Service	2 548	109
7 Total for proposed system (3-6)	12 147	438

Table 4: Overview of the ROM and RAM overhead of the implementation

Energy cost: this paper provides a theoretical approximation of energy usage based on device specification spreadsheets [17]. The estimated transmission energy overhead is 1.6µJ per sent or received byte, and 40 µJ per millisecond of execution time. This gives the following energy overheads: (1) **Token user installation**: transmission of 72B token and one 32B decryption operation: 1.6*72 + 4.64*40 = 300.8 µJ. (2) **Processing a secured message**: 14B transmission overhead and one decryption of 32B: 1.6*14 + 4.64*40 = 208 µJ. These figures are not entered into the table due to lack of comparable data.

While the overhead of the access control system is significant, it is still within the limits of a memory constrained device. This shows the feasibility of multi-user management of wireless sensor nodes, allowing direct multi-user access.

5.4 Threat analysis

This section discusses the threats posed by Networked Attackers (NAs) and Physical Attackers (PAs). It also briefly discusses privacy threats to data.

Authorised access is ensured because NAs cannot send authentic reconfiguration requests to a protected service since that requires that the attacker can create a valid MAC. This paper assumes that the only way to forge such a MAC is by breaking the secret key, which with a sufficiently strong algorithm should be computationally impossible. **Integrity of messages** is ensured because NAs cannot modify messages which are afterwards accepted by the access control infrastructure, since he cannot create a valid MAC. **Freshness of messages** is ensure by the authenticated timestamp attached to the service request, and by the timestamp and timeout value in the user deployment token. The infrastructure thus ensures that an NA cannot access protected services nor interfere with requests other then preventing delivery.

An NA can perform a **denial of service attack**: jamming the networking layer, eliminating the possibility to send data to the node and taxing its resources. This paper does not directly address such attacks. However the system can be augmented with the following mitigation strategies: 1) temporarily disable incoming communication when an invalid message is received, preventing further resource usage, and 2) sending a notification to the PO when one or more invalid messages are detected, signalling a configuration error or attack and allowing the PO to initiate other actions.

Physical attackers can perform a **node capture attack**, revealing all key material from that node. Preventing this attack requires either costly protected hardware modules, or the physical security of the sensor nodes. When these countermeasure are not or cannot be used, a physical attacker can probe any node and gain complete access to all node information. Using the proposed framework he has no access to any other nodes due to the fact that all keys are node unique. However, if a user uses the same key for multiple nodes to allow for group reconfiguration, then the other nodes are vulnerable too. This attack however can be recovered from or prevented. Since each user can be deleted by an administrator user, which can always be added using the node's secret key, the node can recover from a user whose key material has been compromised. Since only the long term secret key of the captured node is known, the PO can always create a new administrator on his nodes and remove the attacking user. The node secret key can only be changed with the secret key, so only the PO can perform a node re-keying. The attack can be prevented by prohibiting group keys: the user has to use a unique key for accessing each node. While this increases overhead, it significantly increases the security of the platforms. It is thus advisable to use this policy for critical node services.

Due to the ubiquitous nature of sensor nodes, it is possible that in certain contexts, the sensors pick up **privacy** sensitive information. It is possible that in such cases, the PO should not perform certain operations or readings because they intrude on the privacy of the tracked subject. However, such attacks are non-technical and concern the legal issues of sensing by POs rather than technical ones. In such cases, these constraints will be entered in the contract that exists between the users and the POs, as stated in section 4.1. The contract will then state that the PO is not allowed to read certain data, which can potentially be translated in user permissions and deployed on the node.

6 Future Work and Conclusion

6.1 Future work

This paper presented a light weight access control system allowing multiple users from different parties to securely share WSN services. However the current architecture has still some possible avenues of future work.

The first avenue is the exploration of the boundaries of access control policy expressiveness. The current implementation allows for basic role based access

control. Access to a service only depends on the user identity and his roles. More advanced ways to perform access control will be investigated where the decision also depends on the arguments of the service call in addition to the role of the requester. For example when performing reconfigurations, it is desirable to be able to restrict the range of values which a user might set, such as the sampling rate of a temperature component.

Secondly, in order to securely share sensor nodes between multiple parties, the execution of requests and configurations of the different parties must be isolated in order to prevent interference. This requires hardware support from the node platforms, which the current generation lacks. Future work could investigate new hardware designs allowing isolated execution of code on sensor nodes.

The third point of future work is the monitoring and auditing of user behaviour. Once a strong authentication and authorisation system is set up, the node can log and audit user behaviour. This auditing can be used to allow charge-back of node usage, enabling POs to recuperate the cost of deploying a wireless sensor network, and monitoring behaviour for malicious use.

6.2 Conclusion

Future sensor networks will operate in dynamic multi-user environments. Multiple users will interact with low resource nodes to retrieve information or modify node configuration. Due to strong resource constraints and unique interaction patterns, traditional access control systems cannot be used. Contemporary research in this area often neglects providing authorisation functionality or adequate management of access control policies.

This paper presented an access control infrastructure for resource constrained wireless sensor nodes. The infrastructure uses a protocol based on symmetrical encryption for authentication and a role based access control framework for authorisation. It provides a management service allowing easy addition, modification and removal of users and roles. The combination of these elements creates a strong access control infrastructure for resource limited WSNs.

The infrastructure is evaluated by means of a prototype which shows the validity of the approach. The overhead of the access control mechanism is significant, but originates mostly from expensive cryptographic algorithms. These algorithms can however be shared with other node systems (secure storage or secure routing), limiting the actual increased cost of the infrastructure. This paper shows that low power micro-controllers can support secure multi-user access control of node services with per user and per party unique access policies.

Acknowledgements

Research partially funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT), the Research Fund KU Leuven, and the EU FP7 project NESSoS. With the financial support from the Prevention of and Fight against Crime Programme of the European Union (B-CCENTRE). Project is conducted in the context of the IWT-SBO-STADiUM project No. 80037.

References

1. Wark, T., Corke, P., Sikka, P., Klingbeil, L., Guo, Y., Crossman, C., Valencia, P., Swain, D., Bishop-Hurley, G.: Transforming agriculture through pervasive wireless sensor networks. Pervasive Computing, IEEE **6**(2) (april-june 2007) 50 –57
2. Zhang, Z., Chen, Q., Bergarp, T., Norman, P., Wikstrom, M., Yan, X., Zheng, L.R.: Wireless sensor networks for logistics and retail. In: INSS, Washington, DC, USA, IEEE Computer Society (17-19 2009) 1 –4
3. Zatout, Y., Campo, E., Llibre, J.F.: Wsn-hm: Energy-efficient wsn for home monitoring. In: ISSNIP, Washington, DC, USA, IEEE Computer Society (dec. 2009) 367 –372
4. Leontiadis, I., Efstratiou, C., Mascolo, C., Crowcroft, J.: Senshare: transforming sensor networks into multi-application sensing infrastructures. In: EWSN'12, Berlin, Heidelberg, Springer-Verlag (2012) 65–81
5. Oliveira, L.B., Kansal, A., Priyantha, B., Goraczko, M., Zhao, F.: Secure-tws: Authenticating node to multi-user communication in shared sensor networks. In: IPSN '09, Washington, DC, USA, IEEE Computer Society (2009) 289–300
6. Huygens, C., Joosen, W.: Federated and shared use of sensor networks through security middleware. In: ITNG '09, Washington, DC, USA, IEEE Computer Society (2009) 1005–1011
7. Priyantha, N.B., Kansal, A., Goraczko, M., Zhao, F.: Tiny web services: design and implementation of interoperable and evolvable sensor networks. In: SenSys '08, New York, NY, USA, ACM (2008) 253–266
8. Dolev, D., Yao, A.: On the security of public key protocols. Information Theory, IEEE Transactions on **29**(2) (1983) 198–208
9. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: Spins: security protocols for sensor networks. Wireless Networking **8**(5) (2002) 521–534
10. Karlof, C., Sastry, N., Wagner, D.: Tinysec: a link layer security architecture for wsns. In: SenSys '04, New York, NY, USA, ACM (2004) 162–175
11. Werner, F., Benenson, Z.: Formally verified authenticated query dissemination in sensor networks. In: SPECTS'09, IEEE Press (2009) 154–161
12. Benenson, Z.: Realizing robust user authentication in sensor networks. In: Real-World Wireless Sensor Networks (REALWSN). (2005)
13. Ren, K., Lou, W., Zhang, Y.: Multi-user broadcast authentication in wireless sensor networks. In: SECON '07. (june 2007) 223 –232
14. He, D., Bu, J., Zhu, S., Chan, S., Chen, C.: Distributed access control with privacy support in wireless sensor networks. Wireless Communications, IEEE Transactions on **10**(10) (october 2011) 3472 –3481
15. Yu, S., Ren, K., Lou, W.: Fdac: Toward fine-grained distributed data access control in wireless sensor networks. Parallel and Distributed Systems, IEEE Transactions on **22**(4) (april 2011) 673 –686
16. Godik, S., Moses, T.: Extensible access control markup language (xacml), v2.0. Technical report, OASIS (2005)
17. Atmel: Avr raven Available as `http://www.atmel.com/tools/AVRRAVEN.aspx`.
18. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: LCN '04, Washington, DC, USA, IEEE Computer Society (2004) 455–462
19. Hughes, D., Thoelen, K., Horré, W., Matthys, N., Cid, J.D., Michiels, S., Huygens, C., Joosen, W., Ueyama, J.: Building wsn applications with looci. In: IJMCMC **2**(4), Hershey, PA 17033, USA, IGI Global (Oct 2010) 38–64