R. Roman, J. Lopez, and P. Najera, "A Cross-layer Approach for Integrating Security Mechanisms in Sensor Networks Architectures", *Wireless Communications and Mobile Computing*, vol. 11, pp. 267-276, 2011.
http://doi.org/10.1002/wcm.1006
NICS Lab. Publications: https://www.nics.uma.es/publications

A Cross-layer Approach for Integrating Security Mechanisms in Sensor Networks Architectures

Rodrigo Roman,Javier Lopez, Pablo Najera
Computer Science Department, University of Malaga, Spain
{roman,jlm,najera}@lcc.uma.es

May 7, 2010

Abstract

The wireless sensor networks (WSN) paradigm is especially vulnerable against external and internal attacks. Therefore, it is necessary to develop security mechanisms and protocols to protect them. These mechanisms must become an integral part of the software architecture and network stack of a sensor node. A question that remains is how to achieve this integration. In this paper we check how both academic and industrial solutions tackle this issue, and we present the concept of a transversal layer, where all the different security mechanisms could be contained. This way, all the elements of the architecture can interact with the security mechanisms, and the security mechanisms can have a holistic point of view of the whole architecture. We discuss the advantages of this approach, and also present how the transversal layer concept was applied to a real middleware architecture.

Keywords. Sensor Networks, Security, Cross-Layer, Middleware, Architecture.

1 Introduction

A wireless sensor network (WSN) can be abstracted as the "skin" of a computer system, an independent "sensory system" capable of obtaining and processing physical information (e.g. temperature, soil mosture) through its "sensory cells" (i.e. sensor nodes). With commercial applications ranging from precision agriculture [1] to healthcare [2], sensor networks are slowly leaving the laboratories and reaching the real world. Nevertheless, the evolution of this paradigm is far from over: there are still advances in this area in terms of complexity, heterogeneity and integration. The elements of a wireless sensor network, the sensor nodes, can be able to interact with complex devices (i.e. actors) that take decisions according to the data received and perform appropriate actions upon the physical world [3]. Sensor nodes can also work in unconventional environments,

such as underwater [4], where their unique characteristics have a great influence over the architecture and the protocols of the network. Finally, sensor nodes can become an integral part of other complex systems, such as embedded peer to peer systems (EP2P) [5], where nodes need to interact with other wireless devices in an ad hoc fashion.

One of the challenges that needs to be closely addressed in these evolved sensor networks is security. As sensor networks are specially vulnerable against external and internal attacks, it is necessary to implement certain security mechanisms and protocols that will either minimize or prevent the malicious effects caused by such attacks. These security mechanisms should be, in most cases, adapted for every specific application, because the requirements of the application and its context will influence over their nature. For example, due to the limited bandwidth efficiency of the sea water medium [4], it is desirable to use protocols that exchange as less bits as possible.

Aside from the development of security mechanisms, there is another important security-related aspect that must be also taken into account: the integration of those mechanisms inside a software architecture and network stack. The components and different layers of a software architecture need of the mechanisms as tools that will help to enforce the security requirements. However, a single type of architecture may not fit the needs of every kind of application. For example, cross-layer architectures seem to suit the specific needs of underwater sensor networks (UWSN), while certain environments (e.g. home automation) with heterogeneous devices can make use of the (non-strict) layered architecture of ZigBee.

The purpose of this article is to analyze how the security mechanisms could be integrated with the existing architecture paradigms (cross-layer and layered), allowing the possibility of adapting or including new security mechanisms without breaking the design. As all these mechanisms share the same goal, which is providing security to other logical parts of the architecture, they could be regarded as a single logical unit within the architecture. Thus, the creation of a transversal layer for security that can be accessed by all the elements of the architecture is considered. Besides, we will explain how the concept of a transversal layer has been successfully applied to a middleware architecture.

The paper is organized as follows. Section 2 reviews the advantages and disadvantages of the existing architecture paradigms (layered and cross-layer) in sensor networks, and section 3 discusses the relationship between these architectures and the security mechanisms. Section 4 introduces the concept of a transversal layer for security, and explains how the different advantages of both layered and cross-layer architectures are retained. Section 5 illustrates the applicability of the transversal layer using the SMEPP middleware architecture as an example. Finally, section 6 concludes the paper.

2 Wireless Sensor Networks Architectures

2.1 Layered and Cross-Layer Approaches

The architecture and protocols of any kind of sensor network are highly dependant on the requirements of the application where it is deployed. For example, in scenarios where the location of the nodes is significant, routing protocols may route the packets using geographical information, while other scenarios (e.g. fire detection) may require protocols that route the packets based on the data obtained from the sensors. Therefore, any software architecture and network stack designed for sensor networks should be flexible and extensible in order to support different types of protocols in different contexts. Whether this kind of architecture should be designed as a rigid layered system or as a completely component-based system with cross-layer interactions is yet to be specified.

Traditionally, network design has followed a layered communication architecture. In this architecture, communication tasks are separated into several layers, with a clear definition of the functionality of each layer. In a layered communication stack, the interaction amongst layers occurs through well-defined standardized interfaces that connect only the neighbouring layers in the stack. By using this approach, it can be possible to provide a high degree of modularity and interoperability amongst heterogeneous networks.

The benefits of using a well-defined layered architecture are numerous. Due to its modularity, designers can easily understand the behaviour of the overall system. It also accelerates the development the design and implementation phases by enabling parallelization of effort. Designers can focus their effort on a particular subsystem with the assurance that the entire system will interoperate. Moreover, individual modules can be upgraded without requiring of a complete system redesign, increasing the longevity of the system. Widely used in wired networks (e.g. the Internet TCP/IP model), this kind of architecture can be also applied to wireless networks, including sensor networks.

However, is precisely in the field of wireless networks where this traditional network stack point of view is being challenged. The unique problems and opportunities created by wireless links, the resource constraints inherent to sensor nodes, and many other design challenges, are enabling the development of novel cross-layer designs [7]. Cross-layer design enables different layers of the communication stack to share state information or to coordinate their actions. For example, supplying information on a node's remaining battery energy to all of the nodes' communication layers.

The use of a cross-layer approach in a sensor network environment yields many advantages. Nodes could be able to manage several performance aspects, such as power management and error notification, that cut across traditional layers. Also, nodes could have access to their current resources and processes (e.g. wireless channel link properties), which contributes favorably to the autonomy and self-configurability of the node (e.g. adapting its behaviour to the local state of the transmission medium). Besides, the inherent constraints of the nodes requires of fine-grained optimization techniques between layers.

There are some aspects that have to be taken into account when designing cross-layer architectures. Cross-layer design opens the floodgate of information flow across layers, raising concerns on multiple, sometimes subtle, interactions amongst existing layers. This makes the architecture more difficult to develop and maintain, as there may be some new dependencies that must be taken into account. Also, the introduction of excessive and uncontrolled interactions can break the design of the system, hindering its usefulness and longevity [6]. As a result, cross-layer designers must consider the impact of their design with a holistic view that includes the long-term development and innovation considerations. The benefits of the different cross-layer design proposals and its possible coexistence should be studied in more deep [7]. Still, as of today, there have been both academic and standard network stacks that implement some cross-layer optimizations in sensor networks.

2.2 Prototypes and Industrial Stacks

The use of cross-layer optimizations and architectures has been widely considered in sensor networks designs and prototypes developed in the academia. In fact, in his seminal paper of 2002, Akyildiz considered the existence of transversal management planes interacting with the layered architecture [8]. Later, he even changed his initial point of view regarding sensor networks architectures and considered that the best solution is to use a unified cross-layer module [9]. Other architectural examples for wireless sensor networks include TinyCubus [10], The "Sensor Protocol" [11], SensorStack [12] and many others. Even more, the "de-facto" standard O.S for sensor networks environments, TinyOS, introduces cross-layer optimizations as a part of its core design [14].

The aspects of a network layer that should be included into a cross-layer design for sensor networks are outlined by all these architectures. First, there are some services that should be accessed by all the layers of the network: power management, network discovery, localization, synchronization, and security. Second, the network architecture should follow a component-based approach, where the functionality that is used by two or more layers (e.g. symmetric key algorithms) should not be replicated. Third, there should exist mechanisms that allow one layer of the network to access the services of non-adjacent layers, for optimization purposes (e.g. an application accessing directly to the hardware services). Finally, it should be required to have certain mediators that provide system information and internal information from other layers through well-defined interfaces.

As for industrial standards, one of the suites of high level communication protocols that have been standardized and applied to some sensor networks deployments is the ZigBee stack [15]. Created in 2004 by the ZigBee Alliance, the major purpose of the ZigBee stack is to enable broad-based deployment of low cost, low power wireless networks that addresses the needs of remote sensing, monitoring and control. In particular, the ZigBee stack is targeting the following applications: Advanced Metering Infrastructure (AMI), Commercial Building Automation (CBA), Home Automation (HA), Personal, Home and Hospital

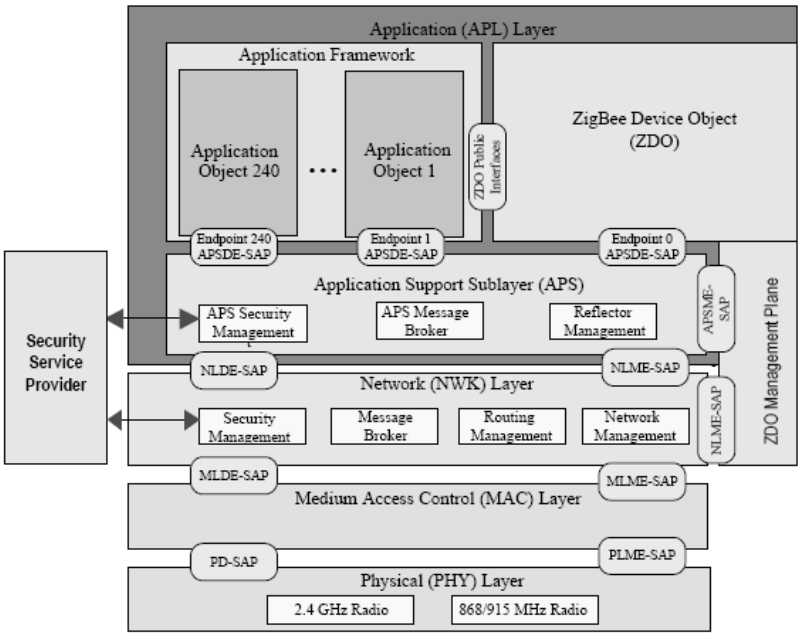

Figure 1: Outline of the ZigBee stack architecture

Care (PHHC), Telecom Applications (TA), and Wireless Sensor Applications (WSA). For wireless sensor networks, as of 2009 the ZigBee Alliance is still defining the profile specification that will allow the creation of sensor networks applications through ZigBee.

Figure 1 shows an overview of the ZigBee network stack, which operates over the IEEE 802.15.4 standard. The network stack is organized into two distinct layers: Network (NWK) layer, and Application (APL) layer. The Network layer allows end-to-end communication and multi-hop capabilities. The Application layer is partitioned into several subcomponents: the Application Support Sublayer, which combines the functionality of a traditional transport layer and of an application layer, the ZigBee Device Object, in charge of storing design parameters and management commands, and the Application Framework, containing the application profiles.

While the ZigBee network stack follows a layered approach, there are some cross-layer interactions. For example, the ZDO Management Plane allows all the ZigBee layers to access the design information and management commands included inside the ZigBee Device Object. Besides from specific optimizations, the ZigBee stack does not consider the existence of certain transversal services that should be used by all layers, such as power management. Also, aside from the management planes, the ZigBee stack follows a strict layered approach, where the application layer cannot have access to the services provided by the link layer.

3 Security and Integrability

3.1 Security Mechanisms for Sensor Networks

In any environment, either physical or logical, there exists the need of maintaining someone or something safe, away from harm. This is the role of security. On any computer-related environment, security can be considered as a non-functional requirement that maintains the overall system usable and reliable, protecting the information and information systems. In fact, in any kind of sensor network, security is of paramount importance: the network must be adequately protected against malicious threats that can affect its functionality. Also, due to the role of sensor networks as "sensory systems" and "actuator systems", any disturbance in a sensor network may have consequences in the real world.

However, achieving this goal is not an easy task, because sensor networks are specially vulnerable against external and internal attacks due to their peculiar characteristics. The devices of the network (i.e. sensor nodes) are highly constrained in terms of computational capabilities, memory, communication bandwidth and battery power. Additionally, it is easy to physically access such nodes: either they must be located near the physical source of the events, or their mobility allows an attacker to easily locate and subvert them. Furthermore, any internal or external device can access to the information exchange because the communication channel is public.

As a result, sensor networks have to face multiple threats that may easily hinder its functionality and nullify the benefits of using its services: communication channel attacks, denial of service, node compromise, impersonation attacks, protocol-specific attacks, etc. It is clear that there is the need of using security mechanisms either to prevent the attacks from influencing over the functionality of the network or to minimize the adverse effects of such attacks [16].

The communication channel can be adequately protected against external attacks by using security primitives, although the security credentials (i.e. secret keys) need to be distributed using key distribution protocols. Support protocols, such as situation awareness mechanisms and intrusion detection systems, can offer a basic level of protection to "core protocols" (routing, aggregation, time synchronization). Note that these "core protocols" also can, and should, implement protection mechanisms of their own. Besides, there are other specific protocols, such as code distribution, that must be able to withstand specific attacks. Finally, it is possible to include other security protocols, such as code attestation (i.e. check whether the code of a node is valid and not tampered), that may help to maintain the overall security of the network.

3.2 Integrating the Security Mechanisms

By means of the security mechanisms, it is possible to create and implement secure services and protocols that fulfill essential security properties such as confidentiality, integrity and authentication. For example, by using security

primitives such as block ciphers and message authentication codes, we can create secure protocols that are able to protect the communication channel between nodes or between peers. Also, as survivability is an important issue in sensor networks, any situation awareness mechanism in charge of monitoring the behaviour of the nodes and their neighbourhoods will provide crucial information for allowing self-configurability in presence of possible problems (e.g. nodes "dying", broken links).

The components and different layers of a software architecture need of these security mechanisms as tools that will help to enforce the security requirements of an application. Therefore, the security mechanisms themselves must become an integral part of the software architecture. A question that remains is where and how to integrate them inside the architecture. It is necessary to maintain the logical coherence in the design of the architecture, while allowing the possibility of either adapting or including certain security mechanisms in order to comply with the requirements of the applications. Besides, due to the limited resources available to most sensor nodes, there should exist certain optimizations such as different services sharing the same primitive.

By including the security services as a well-defined layer within a layered architecture, we can benefit from the advantages of this approach: modularity, interoperability assurance, and localized design and upgrading. Still, there are situations where the functionality of a layer would need to be replicated within the architecture. For example, an end-to-end secure channel located at the application layer [17] can use the same cryptographic mechanisms of a secure link layer communication channel [18]. Besides, there are other situations where a service need to access information contained in other layers in order to work correctly. For example, in order to know the actual state of a node and its neighbourhood, it is necessary to analyze variables such as actual battery levels, packet signal strength, and packet headers. Moreover, a radio fingerprint mechanism in the physical layer [19] can be used by other layers to authenticate the source of a message and detect attacks such as replication attacks.

All these concerns have been considered by prototypes from the academia and industrial standards, and most of them suggest some cross-layer optimizations for integrating the security mechanisms as part of an architecture. It was already pointed out by Srivastava [7] that security related issues need to be handled across the layers in a holistic manner. This point of view is shared by most designs coming from the academia, such as the "sensor protocol" [11] and SensorStack [12]. In particular, the "sensor protocol" considers that security should be encapsulated in a single layer, not distributed amongst layers. However, that architecture provides few discussions about this particular issue.

Regarding the ZigBee stack, the need to optimize the resource usage in the implementation of the security mechanisms is acknowledged by the definition of a specific component, the Security Service Provider. This component is shared by the Network Layer and the Application Support Sublayer, and its main task is to provide security mechanisms for data encryption. However, the architecture does not provide support for other security services, such as intrusion detection and self-healing systems, that need a holistic point of view of the

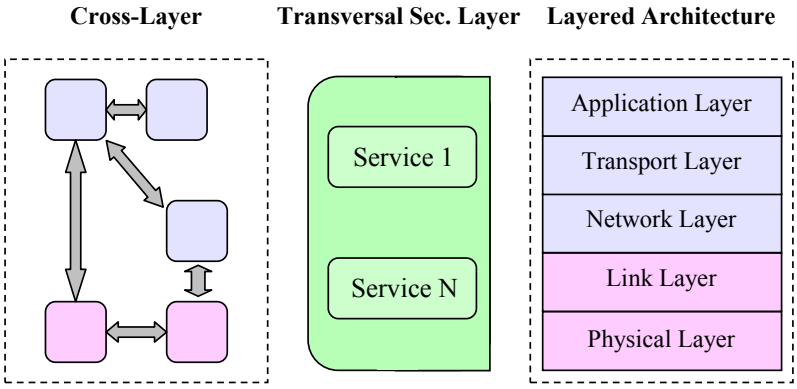

Figure 2: Security as a transversal layer

whole architecture.

From the previous discussions, it would seem clear that in a sensor network context the security mechanisms and services need to interact with the existing layers of an architecture in unconventional ways, by considering cross-layer approaches. Nevertheless, as both cross-layer architectures and layered architectures are to be found in the real world, it is necessary to evaluate how the security mechanisms could be satisfactorily integrated with any type of architecture without endangering the consistency of its design.

4 Security as a Transversal Layer

The major purpose of the security mechanisms is usually not to implement the logic of the application, but to offer support for the creation of secure applications. We can see then the security mechanisms as tools, used either by other elements of the architecture or by other security mechanisms. As they share the same goal, they can be considered as a single logical unit inside the architecture, thus we could group all the security mechanisms within a single layer. By using this approach, we can have a better management of all the mechanisms, delimiting their functionality and controlling their interdependencies. As for the relationship between the mechanisms and the rest of the architecture, the mechanisms should be able to access all the elements of the architecture in order to obtain information, and the architecture itself should be able to access any of the security services when needed.

Therefore, security should be considered as a transversal layer (cf. Figure 2), that in both layered and cross-layer architectures will be able to provide information and services to the other elements of the architecture. Security services and primitives (e.g. protocols, algorithms) are located inside that transversal layer, and they interact between them and between the other elements of the architecture through well-defined interfaces. Besides, the transversal layer only

provides security-related services, invokes the services of other layers in order to obtain information, and signals specific events to warn other layers about a specific situation. Therefore, any layer cannot access the functionality of other layers through the transversal layer in an indirect way.

From a technical point of view, the interfaces provided by the transversal layer must follow these two principles: independence and extensibility. The specific implementation details of the algorithms and protocols should not affect the definition of the interfaces, so the components that use the transversal layer do not need to change their implementation whenever a existing primitive is upgraded or changed. This can be achieved by defining a common interface for services and primitives of the same type (e.g. symmetric cryptography primitives), and also by using mechanisms such as the "factory" design pattern (in a similar fashion to the Java Security API [13]). As similar primitives and services will use the same type of interface, it is simple to add a new element whenever it is needed.

Precisely, the specific security elements that are contained within the transversal layer can be chosen according to the necessities of the applications and the capabilities of the devices. For example, if a specific application employs of digital signatures, the transversal layer can provide an interface for accessing a signature service. Such service can be implemented by using lightweight algorithms (e.g. signature algorithms based on Elliptic Curve Cryptography) if the devices are constrained in terms of resources. More powerful devices can implement more algorithms (e.g. identity-based signatures) inside the transversal layer. The transversal layer should also provide the services and primitives it contains whenever it is queried, so it is possible for two devices to negotiate security elements that they have in common.

Note that the definition of wrappers that allow any component of the transversal layer to access the functionality of elements located outside it is also important. Any change in the interface of the architecture will impact on the wrapper, but not on the implementation of the security elements located inside the transversal layer. In addition, if we do not define interfaces that access functional parts of the elements of the architecture (such as sending messages through a communication channel), then it will not be possible to bypass the layered structure of an architecture through the transversal layer.

4.1 Benefits and Applicability of Transversal Layers

By using the transversal layer approach, it can be possible to retain most of the benefits of a layered architecture. All the security mechanisms and services are contained within the transversal layer, thus the modularity of the system is preserved. Besides, the transversal layer should only be accessed through well-defined interfaces, thus any component/layer of the system, located either inside or outside the transversal layer, will know how to interoperate with them in advance. In addition, since the transversal layer is isolated from the other layers, and even the internal components of the transversal layer can only access other components through their interfaces, it can be possible to either modify

of upgrade the internal design of the layer without interfering with the other elements of the architecture. Note also that a layer cannot use the transversal layer as a bridge to access the services of unconnected layers, thus there is no risk of hidden dependencies.

The benefits of cross-layer architectures are also maintained by the transversal layer. The services contained within the transversal layer can access information published by the services of other layers, thus can have a holistic point of view of the state of the device. For example, a situation awareness service, in charge of monitoring the actual state of a node and its neighbourhood, can retrieve node status information from other layers. The external layers can also benefit from this transversal approach in many ways. First, since there is just one instance of a security mechanism (e.g. a cryptographic primitive), there is no need to replicate its functionality in order to implement different protection mechanisms (e.g. link layer protection and end to end protection). Second, all security mechanisms and services can be accessed by all the layers of the architecture. As a result, it can be possible to implement more effective protection procedures. For example, if an intrusion detection system is included in the architecture, both applications and communication protocols can use its outputs to limit any interaction with rogue nodes.

Finally, the possible disadvantages that may exist whenever cross-layer relationships are included in an architecture are reduced when using the transversal layer. The information flow between the transversal layer and other layers is delimited by well-defined interfaces, thus interactions amongst existing layers are no longer subtle. Any change inside the elements of the transversal layer must adhere to the definition of the interfaces, thus the chances of breaking the design of the whole architecture are slim. This level of isolation also limits the possible dependencies that may appear between layers after a change takes place.

Not only the transversal layer retains the benefits of both layered and cross-layer architectures, but also it can be integrated within both types. In layered communication architectures, the existence of a transversal layer containing the security services will allow the modification of those services according to the requirements of the applications (e.g. choose a different security primitive) without causing collateral effects that may negatively affect the other layers. In addition, it can be possible to add new security services derived from the necessities of the applications and / or long-term academic research.

On the other hand, in cross-layer architectures, the security layer behaves as another component of the architecture that can be accessed from any other component, proving security-related services and information. By centralizing all security services inside one single component, we can improve the overall stability and maintainability of the architecture, and also we can provide a better control over the possible interactions and dependencies that may arise.

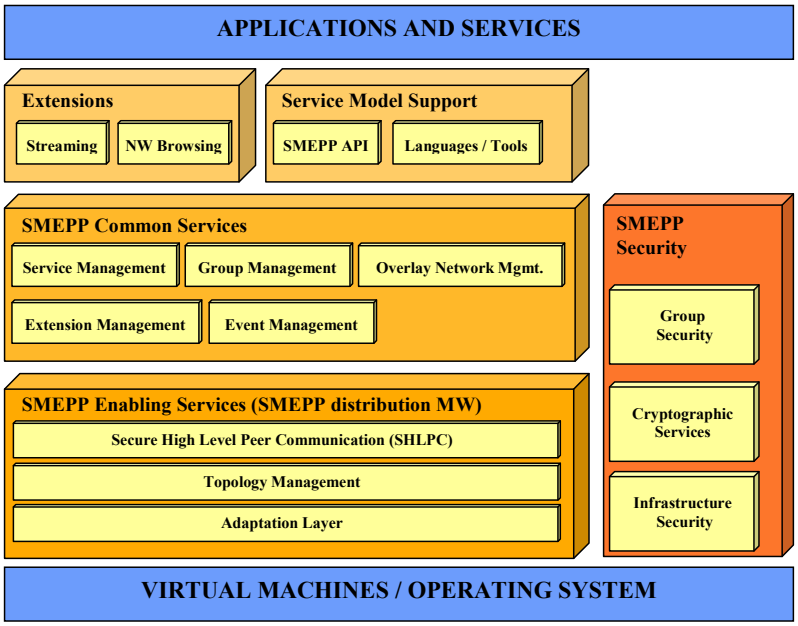

Figure 3: An Overview of the SMEPP functional structure

5 Implementing transversal layers: The SMEPP Architecture

We have applied the concept of a transversal layer containing security mechanisms to a complex middleware architecture, developed under the European project SMEPP (Secure Middleware for Embedded Peer-to-Peer systems, FP6-IST-033563) [20]. The main purpose of this middleware is to allow the development of secure applications for different types of embedded devices interacting in a P2P fashion without any pre-existing infrastructures. This particular paradigm is known as Embedded Peer-to-Peer (EP2P) system.

The general SMEPP Middleware architecture is shown in Figure 3. It is divided into three functional layers. The top layer consists of the Domain-specific Middleware services and the Service Model Support that provide the actual API for application developers for using the SMEPP Middleware. The second layer from the top is the Common Middleware Services that provides the actual functionality of the Middleware defined by the service model. Finally, the bottom layer is the distribution middleware that contains all the framework implementations needed by the Middleware services for providing the required functionalities. On sensor networks, the architecture is simplified to contain only those components that are useful to obtaining and processing data in a local environment, and it is known as SMEPPLight. This lightweight architecture retains the components related to Groups, Events, Topology, and Security.

Regarding security, in both SMEPP and its lightweight implementation SMEPPLight it is considered as a transversal layer that can be accessed by all the services of the middleware. This point of view coincides with the approach presented in section 4. The principal security components in this architecture contained within the transversal layer are the following:

- *Group Security* is concerned with the security related aspects of group establishment and maintenance. In SMEPP, the members of the network can join certain groups, and they can access to the services of other peers only if they belong to the same group. This component is responsible for both the authentication of peers which want to join these groups and the secure communication amongst group members. As this component is isolated from the actual component that manages the groups, it is possible to implement different levels of security (e.g. joining a group using either symmetric keys or public/private key pairs) without modifying the entire architecture, and different groups can have a different level of protection according to their own security requirements. Besides,this component does not send any information through the network, thus there is no hidden interdependencies with the lower layers.

- *Cryptographic Services* provide common functionality which is required by other components (e.g. Secure Topology Management, Group Security). This includes cryptographic primitives (encryption/decryption, digital signatures, Message Authentication Codes, unkeyed hash functions) as well as supporting functions (random number generation, key storage, certificate management). As there is one single instance of every primitive, located within this component, there is no need to replicate their functionality over all the architecture. Besides, since the services offered by this component are provided through well-defined interfaces, it is a simple task to replace the primitives located within the component.

- *Infrastructure Security* encompasses device-specific support for security functionality which facilitates the implementation of the security-related components. An important example are secure instruction sets (instruction set extensions for cryptography), which speed up cryptographic processing. These features can be used to realize the Cryptographic Services more efficiently, which in turn reduces the overhead incurred in all security-related components. This component needs to be rewritten for every specific device where the middleware is implemented, but any changes will not affect the other components of the architecture.

As an example of the relationship between this security layer and other services, we can mention the Group Security component. Its services are used by the SMEPP Common Services layer (Group Management component, authenticates a certain peer) and by the SMEPP Distribution MW layer (Secure High Level Peer Communication component, protects the messages of a group). Regarding the interaction between components inside the same security layer, the

Cryptographic Services component will use the cryptography extensions located in the Infrastructure Security component if available. If the design is changed and new cryptographic extensions are available, there is no need to change the whole architecture: the Cryptographic Services component will simply use the new extensions.

5.1 Implementation and Prototypes

The SMEPP middleware has been successfully implemented and tested by the SMEPP consortium in both high-end devices (PCs), PDA-like devices, and sensor devices. For high-end devices we have used Java SE technology, while for PDA-like devices we have used Java ME technology with the Connected Device Configuration (CDC) framework. On sensor networks, we have used nesC over TinyOS, and the memory footprint of the SMEPPLight middleware (56KB RAM, 2944B ROM) is good enough to implement secure applications for MICAz-class nodes (128KB RAM, 4-8KB ROM).

Both the Java implementation (through the UM-RTCOM component model [21]) and the nesC implementation use component-oriented programming, thus the interfaces of the transversal layer can be appropriately defined. The interfaces of the different primitives and protocols of the transversal layer are defined independently of their specific implementations, allowing extensibility and reusability. For example, all the different security levels (no security, authentication through shared keys, authentication through public key cryptography) of the *Group Security* component are implemented using different protocols (the ISO/IEC 9798-2 standard [22] and a modified Needham-Schroeder-Lowe protocol [23]), but are provided through a single `negotiationStep()` method. This way, in order to join a group, the *Group Management* component can choose a certain security level without changing its implementation logic.

The mechanisms that are located inside the transversal layer are also adapted to the capabilities of the different devices. For example, the SMEPP middleware includes all the different security levels and primitives implemented, while the SMEPPLight version of the middleware only support group authentication through shared keys and support for public key cryptography can be excluded. Note that if a specific application does not need to use a security level, such level can be deleted at design time, thus reducing the size of the overall middleware.

The consortium have also created various prototypes to test the viability of the SMEPP middleware as a framework for the development of secure EP2P applications. Those prototypes are digital home services with mobile devices and nuclear power plant monitoring [24]. Our digital home prototype is focused on elderly care, where either the user or the system itself can raise an alarm when an exceptional situation arises. Using the group capabilities of SMEPP, it is possible to discover an user (e.g. a member of the family) that can accept the alarm. Once a communication channel is established, both users can securely exchange text messages, images, and/or video streams.

As for nuclear power plant monitoring, we concentrate on Dosimetric Control and Environmental Radiological Control. A network of operators wearing

SMEPP-enabled dosimeters can provide real-time information on the radiation levels of the power plant, and the EP2P capabilities of the network allows an efficient response if any operator gets exposed to harmful levels of radiation. Besides, in case of an emergency, more devices can be scattered to get an accurate information on the state of the power plant. All the interactions between the devices are protected by using the security capabilities of the SMEPP middleware.

6 Conclusions

The integration of security mechanisms and protocols inside the software architecture and network stack of a sensor network is not a trivial issue. In this article, we have presented the concept of a transversal layer, where all the different security mechanisms are contained. This way, all the elements of an architecture can interact with the security mechanisms, and the security mechanisms can have a holistic point of view of the whole architecture. In addition, this concept of a transversal layer has been successfully implemented and tested in a middleware architecture developed under the European project SMEPP [20].

The transversal layer retains the benefits of layered architectures (modularity, interoperability, design longevity), while the advantages of cross-layer architectures (optimization, tunable design) are maintained and its disadvantages (hidden dependencies, poor design) are adequately controlled. Note that the transversal layer approach has all these benefits because the major purpose of the security mechanisms is to provide services to the different elements of the architecture, and they only access other elements either to obtain information regarding the state of the device or to signal an event. Future research includes analyzing how this transversal layer concept could help on optimizing the provisioning of security when using specific sensor network protocols such as 6lowpan [25].

A Acknowledgements

This work has been partially supported by the European Union through the SMEPP (EU-FP6-IST 0333563) project and by the Spanish Ministry of Science and Innovation through the ARES (CSD2007-00004) and SPRINT (TIN2009-09237) projects. The latter is cofinanced by FEDER (European Regional Development Fund).

References

[1] Crossbow Technology, Inc. eKo Pro Precision Agriculture. `http://www.xbow.com/eko/`. Accessed February 20th, 2009.

[2] Cadi Scientific Pte Ltd. CADI SmartSense^{TM}. http://www.cadi.com.sg. Accessed February 20th, 2009.

[3] Melodia T, Pompili D, Gungor VC, Akyildiz IF. Communication and Coordination in Wireless Sensor and Actor Networks. IEEE Transactions on Mobile Computing 6(10) (2007) 1116-1129.

[4] Lanbo L, Shengli Z, Jun-Hong C. Prospects and problems of wireless communication for underwater sensor networks, Wireless Communications and Mobile Computing 8(8) (2008) 977-994.

[5] Brogi A, Popescu R, Gutierrez F, Lopez P, Pimentel E. A Service-Oriented Model for Embedded Peer-to-Peer Systems, Electronic Notes in Theoretical Computer Science 194(4) (2008) 5-22.

[6] Kawadia V, Kumar PR. A Cautionary Perspective on Cross-layer Design. IEEE Wireless Communications 12(1) (2005) 3-11.

[7] Srivastava V, Motani M. Cross-Layer Design: A Survey and the Road Ahead, IEEE Communications Magazine 43(12) (2005) 112-119.

[8] Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey, Computer Networks: The International Journal of Computer and Telecommunications Networking 38(4) (2002) 393-422.

[9] Akyildiz IF, Vuran MC, Akan OB. A Cross-Layer Protocol for Wireless Sensor Networks, Proceedings of the 40th Annual Conference on Information Sciences and Systems (2006) 1102-1107.

[10] Marron PJ, Minder D, Lachenmann A, Rothermel K. TinyCubus: An Adaptive Cross-Layer Framework for Sensor Networks, IT Information Technology journal 47(2) (2005) 87-97.

[11] Culler D, Dutta P, Tien Ee C, Fonseca R, Hui J, Levis P, Polastre J, Shenker S, Stoica I, Tolle G, Zhao G. Towards a Sensor Network Architecture: Lowering the Waistline, Proceedings of the 10th Workshop on Hot Topics in Operating Systems - HotOS X (2005) 139-144.

[12] Kumar R. Adaptable Protocol Stack Architecture for Future Sensor Networks, PhD Thesis, Georgia Institute of Technology (2006).

[13] Java Cryptography Architecture (JCA) Reference Guide. http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html. Accessed February 20th, 2009.

[14] Levis P, Madden S, Polastre J, Szewczyk R, Whitehouse K, Woo A, Gay D, Hill J, Welsh M, Brewer E, Culler D. TinyOS: An Operating System for Sensor Networks, in: W. Weber, J. Rabaey and E. Aarts (Eds.), Ambient Intelligence, Springer Berlin Heidelberg, 2005, pp. 115-148.

[15] ZigBee Alliance. ZigBee Specification.
http://www.zigbee.org/. Accessed February 20th, 2009.

[16] Walters JP, Liang Z, Shi W, Chaudhary V. Wireless Sensor Network Security: A Survey, in: Y. Xiao (Ed.), Security in Distributed, Grid, and Pervasive Computing, Auerbach Publications, CRC Press, 2006, pp. 367-410.

[17] Chung A, Roedig U, DHB-KEY: An Efficient Key Distribution Scheme for Wireless Sensor Networks, Proceedings of the 4th IEEE International Workshop on Wireless and Sensor Networks Security - WSNS2008 (2008) 840-846.

[18] Luk M, Mezzour G, Perrig A, Gligor V. MiniSec: a secure sensor network communication architecture, Proceedings of the 6th International Conference on Information Processing in Sensor Networks - IPSN'07 (2007) 479-488.

[19] Bonne Rasmussen K, Capkun S. Implications of radio fingerprinting on the security of sensor networks, Proceedings of the 3rd International Conference on Security and Privacy in Communications Networks - SecureComm'07 (2007) 331-340.

[20] SMEPP Consortium. Secure Middleware for Embedded Peer-to-Peer Systems, FP6-IST-033563.
http://www.smepp.org/. Accessed February 20th, 2009.

[21] Diaz M, Garrido D, Llopis L, Rus F, Troya JM. UM-RTCOM: An analyzable component model for real-time distributed systems, Journal of Systems and Software 81(5) (2008) 709-726.

[22] International Organization for Standardization, ISO/IEC 9798-2:2008 (Mechanisms using symmetric encipherment algorithms), 2008.

[23] Lowe G. Breaking and fixing the Needham-Schroeder public-key protocol using FDR, Proceedings of the 2nd Intern. Conf. Tools and Algorithms for the Construction and Analysis of Systems - TACAS (1996), 147-166.

[24] Crossbow Solutions Blog. RadMote - Mobile Framework for Radiation Monitoring.
http://blog.xbow.com/xblog/2007/12/radmote---mobil.html.
Accessed February 20th, 2009.

[25] Kushalnagar N, Montenegro G, Schumacher C. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs), RFC 4919, August 2007.