System-Level Design of a TinySec-based Secure Low-Power Protocol for a Remote Meter Reader

Daniel Spanagel and Mauro Prevostini

ALaRI, University of Lugano
CH-6904, Lugano, Switzerland

January 18, 2006

Abstract. New technologies enable to perform standardized processes in a more efficient way. This applies also to the sector of meter reading, where the consumption data of gas, water and electricity had been read manually so far. This process can be dramatically optimized through the usage of new technology by upgrading installed meters with a non-invasive low-power/low-cost optical system using a digital retina[1]. Data can be read through the glass of the meters and transmitted to the utility provider using a wireless network. So, the process of reading manually the meters and collecting the consumption data can be replaced by a more cost efficient solution. This new solution arises some issues on security aspects related to data confidentiality and integrity. The paper will propose a RF protocol based on TinySec in order to satisfy the remote meter reading system requirements taking care of security and low-power aspects.

1 Introduction

Remote Meter Readers (RMR) are devices, which allow the reading of gas, water and electricity consumption data remotely, without needing physical access or a visual inspection of the meter. This kind of meter reading system, depending on the device, can be on demand or in given time intervals. For the transmission of the consumption data there are all the option to do it by wire or wireless, with all the variety of different data transmission protocols currently available. In this paper we will consider the case where a low-power/low-cost optical module is mounted onto an installed meter which does not provide any communication interface and is able to acquire the picture of the consumed data through the glass of the meter, process it and convert it into a digital number[1]. Then this number will be stored in a local buffer and once a day sent to the utility provider. This kind of solution must be based on a low-power protocol because the goal is to install the RMR and never touch it, unless necessary, for years, so that, battery consumption must be very low in order to guarantee a relatively long life time of the devices, usually about 10 years. Besides low-power requirements another very important aspect is security. Data transferred to the service provider

[1] This solution is adopted by a Swiss company called Xemtec [1]

must satisfy confidentiality and integrity requirements. So that the low-power transmission protocol have to be designed taking care of authentication and integrity check mechanisms. The paper describes in section 2 the state-of-the-art of low-power protocols available and in section 3 shows in details which is the problem we would like to solve. In section 4 we will focus on the design of our protocol. Section 5 concludes the paper discussing as well as future works. Section 6 and 7 are dedicated to the acknowledgments and references.

2 State-of-the-art

In this state of the art, the focus will lay in the wireless transmission protocols, strategies and encryption algorithms that are as power efficient as possible, since this is one of the key requirements of the Automatic Meter Reading community (www.amra-intl.org). The data transmission, of course, has to be secure, as customers of a RMR system will not be comfortable if the privacy of their consumption data is not guaranteed.

2.1 Wireless secure communication protocols

There are very well known protocols for wireless communications, like the established WiFi IEEE 802.11 or the Bluetooth 802.15.1 [6] standards. But both standards are not really tailored to embedded systems with low power constraints and not to intensive bandwidth needs, like it is the case for sensor and control devices and especially for our case study device. So is 802.11 mainly designed for mobile desktops or computational intensive devices like PDAs that should achieve high data rates for web surfing, voice and video transmission, etc. Bluetooth addresses mid range data rates, mainly for wire free communication between e.g. office devices that have power supply or for devices that can be recharged often. However a new standard, Zigbee™ [7] that defines network, security, and application framework protocol software and is designed to work on top of the IEEE 802.15.4 PHY/MAC layer standard [2], is targeting control and sensor devices that don't need high bandwidth but need very low energy consumption for long battery lives. E.g. a typical ZigBee energy consumption in standby modus (which should be used the predominant part of the time) would be 3 μA instead of 200 μA in the case of Bluetooth or even 20 mA for WiFi [4]. ZigBee's protocol code stack is estimated to have about 1/4th of the size of Bluetooth's or 802.11's [3], what also makes it better suited for low cost embedded systems with less ROM. The problem of the ZigBee standard is that it is just now released and solutions developed according to the standard are slowly starting to come out in the market. For further and more detailed information about the Zigbee standard please refer to [2],[3] and [4]. Other protocols not mentioned in this section which could be also relevant in future for embedded systems like the one of our case study are protocols developed from the T-Engine [5] standardization forum in Japan.

2.2 Proprietary protocols

Another possibility would be to use proprietary implemented stacks like e.g. the WiSE™ protocol [8]. The advance of proprietary solutions for this application field is that in this early stage of standardization they provide technical advantages, like higher reach ability, lower cost, smaller software stack, etc. But on the other hand, on a longer time horizon, through the standard there will be other advances derived from the scalability, like the possibility of product interoperability, vendor independence, better development tools and in the end, a more convenient price.

2.3 Protocols from the Wireless Sensor Networks field

There is a big research effort in the areas of low power secure protocols, mainly related to wireless sensor networks (WSNs). So there is a huge amount of different power efficient network protocols and approaches for wireless networks, that target different layers of the traditional OSI protocol stack [13]. For security there are also different propositions for energy efficient security link-layers [15], like SPINS [10] or TinySec [11]. In this section we will look closer at this two protocols, since they are the most prospective and our later designed security protocol will rely mainly on the TinySec link-layer.

2.3.1 SNEP

SNEP has been developed into the framework of the SPINS (Security Protocols for Sensor Networks) research project [9] which is a suite of security building block optimized for resource constrained environments and wireless communication. SNEP is a security protocol that provides following security primitives: Data confidentiality, data authentication and data freshness. The approach is targeting a static homogeneous wireless network, where a central base station acts as the central point of trust. Further details of SPINS and the SNEP protocol can be found at [10].

2.3.2 TinySec

TinySec [11] is a fully-implemented link layer security architecture for wireless sensor networks that is part of the official TinyOS release [12]. As SNEP, it is also dealing with constraints like weak processors, small memories and limited energy. TinySec is a further development of SNEP and was released in 2004. It also provides authentication, message integrity, and confidentiality through semantic security. TinySec doesn't provide replay protection as in the case of SNEP, because their developers argue that this characteristic belongs to the application layer. In TinySec, there are mainly 2 security primitives applied:

- **Message Authentication Codes (MACs)** which are commonly used for archiving authentication and integrity and require a secret shared key between the two communication instances.
- **Initialisation Vectors (IV)** which are commonly used to achieve semantic security so that the encryption of the same plain text results in different cipher text as each IV is unique.

There are two security versions of TinySec which can be seen in [11]

- **TinySec-AE:** which provides authentication and encryption. Here the data payload is encrypted and the packet is authenticated with an MAC
- **TinySec-Auth:** this version provides authentication only through a MAC.

2.4 Encryption algorithms optimized for Wireless Sensor Networks (WSN)

So far, in the WSN community, only symmetric algorithms have been implemented, as asymmetric cryptographic algorithms are too computational intensive and add intensive communication overhead [9]. In the last year, there has been done some work on public key cryptography algorithms for WSN [14], [17], but for having a mature working implementation it is still too early. This is why in section 4 we will refer to some symmetric cryptography algorithms that fit well to WSN as they require a good trade-off between security and power effectiveness as well as to the very limited resources of a WSN node. The concepts and results of this field can be easily applied to the present case study, as the resources of the RMR should be very limited and the power constraints are high too, as the RMR should live for a long time (ideally 10 years) with the same battery.

3 Problem description

In this work, the focus is on a case study of the RMR product offered by Xemtec [1]. We choose this product for our case study, because from our point of view it is the most cost effective automated meter reading solution available in the market today. Considering this low power, low cost meter reader we have been inspired to model a low-power protocol for this kind of application. The corresponding RMR is one of the RMR's that is designed as an add-on module to existing, installed meters. The goal is to upgrade non-communicating meters into smart devices in order to avoid a complex and expensive installation of new digital meters. In this peculiar case, this remote meter will be very easy to install, only needing to clip the RMR to the installed meter. The meter reader will have a long lasting battery that thanks to the low power operation of the RMR has to last for approximately 10 years. The architecture that has been chosen for this case study is shown in Figure 1. We can see that the classical analogue meters are complemented by the remote meter reader (RMR), which reads the consumption data and sends it through a RF connection to a data collector (DC). The DC collects the data from the different sources and retransmits it through a more powerful wired or wireless connection to the service provider (SP) which can then use the data for billing, dynamic cost models, etc.

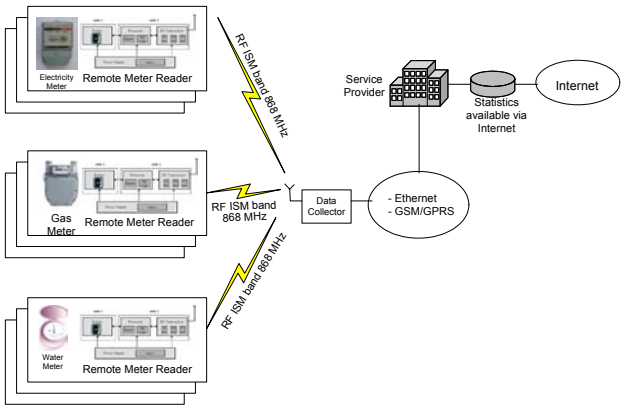

Fig. 1. Architecture of the RMR System

3.1 Functional System Requirements

The RMR should be able to store the recognized consumption data of at least one day, considering the measurement intervals of one every 15 minutes. The RMR should also be able to transmit the stored data over a wireless connection to a DC, if requested by a DC or if triggered by an intern timer scheduled before by a DC. The data transmission between the RMR and the DC should be performed wireless over the license-free frequency bands of 433 and 868/915 MHz. As this frequency bands are open, collisions and loss of transmitted data has to be considered and avoided. It has to be considered as well, that the communication process between RMR and DC is the most power consuming one. In order to minimize the power consumption, a low power communication protocol that also guarantees a safe transmission of data has to be used. The data transmission between the DC and SP can be performed wireless or wired. In the transmission from RMR to DC, data integrity, authentication and confidentiality should be considered, as there is sensitive information to be transmitted. In order to make a low power consumption strategy possible, where e.g. the RMR is in sleep mode almost all the time with exception of short intervals when it wakes up to see if there is a communication request from the DC, the RMR and the DC have to be synchronized. Data integrity guarantees that transmitted or stored data has not been accidentally or maliciously altered. In circumstances where integrity of data is important, hash functions and integrity controls (e.g. MACs) should be considered. The RMR System has to guarantee the integrity of all the sensitive data exchanged. Each Meter Reader needs a unique ID in order to grant the right identification of the RMR. Authentication must be implemented into the system, in order to guarantee that every system module that sends and receives information is really the one who claims to be. Data Collector has to be sure that all received data have really been generated and transmitted by an authorized Remote Meter Reader. Service Provider has to be sure that all received data have really been generated and transmitted by an authorized Data Collector.

4 Protocol Design

In this paper we will discuss more closely a protocol at a deeper level, which grants the data integrity, authentication and the confidentiality of the transmitted data on the wireless link DC⇔RMR. We will not focus on the DC⇔SP link, because in this link doesn't make sense to apply TinySec, in fact here e.g. a 802.11, a fiber optic or an Ethernet network should be applied. The secure communication approach will be mainly based on the before presented TinySec, as it is a link-layer which accomplish all established requirements and because there is an implemented, royalty free stack of TinySec. In this section TinySec is adapted in some extend to the RMR case study, e.g. through low power communications strategies, but the idea is to apply TinySec as close as possible, adopting also the in [11] proposed packet configuration.

4.1 Authentication

Authentication must be implemented in order to guarantee that all data received by DC have really been generated and transmitted by the assumed RMR, to not link erroneously billing information to another service consumer. To this purpose the authentication algorithm makes use of an also at the manufacturing process installed unique key for each RMR. This key, which will be available also for the DC, will be used to calculate a message authentication code (MAC) which will use a cipher block chaining construction (CBC). The DC will be able to relate the corresponding key with the RMR through the RMR ID. Through the MAC the "identity" of the two involved parties is guaranteed, as each party signs their data packages with this key.

4.2 Data Integrity

Normally, to detect transmission errors, a cycle redundancy check (CRC) is computed over the package. Instead, here the CRC is replaced by the MAC. The MAC protects the entire packet data from tampering, but also from redirecting a packet intended for one node to another node, packet truncation, etc. Since MACs detect these malicious changes, they also detect errors in the transmission, which is the reason why a CRC is unnecessary.

4.3 Confidentiality

Confidentiality is achieved through semantically secure encryption, which requires typically an encryption scheme and specifying an initialization vector (IV) format. The last is needed for avoiding that the same plaintext sent twice results in two equal plaintexts and makes possible so the required semantic security. The encryption scheme used will be a symmetrical one, because it is faster than an asymmetrical so that it consumes less power. Nevertheless it allows a sufficient security level for this

application. More specifically, it will be a block cipher scheme, which is a keyed pseudorandom permutation over small bit strings, typically 8 or 16 bits [11]. The standard symmetric block cipher encryption scheme for TinySec is Skipjack[2] and is also proposed here. For this encryption scheme there is a Key needed. We propose also to use a key configured at manufacturing which is unique for each RMR. Consequently each DC⇔ RMR connection will have an own secure link (a so called per-link keying), which for the purpose of the RMR system will offer more than enough security. Even if an adversary get a RMR and is able to extract the key through direct manipulation, this will be the only link he will be able to eavesdrop. Encryption in TinySec is an option. Also for this case study we will let the possibility open at this design stage, if encryption should be applied. This will mainly depend on the specific target system and the requested security constraints. In the following diagrams encryption will be always present for completeness, but of course the one or two encryption steps can be taken out, if not required.

4.4 UML Object model diagrams

In this section a UML [18] object oriented design for the implementation of the RMR secure communication protocol is presented. In this case, the design focuses on a synchronized pull mode.

4.4.1 Data Collector

The object model diagram of the data collector, which can be seen in Figure 2, is composed by the following classes:

- Class `SystemMgr` represents the overall controller in the DC, and it operates over the other classes, e.g. enabling the synchronisation, requesting consumption data from the RMRs, storing the consumption data from different consumers and other necessary functionalities. In order to do this, it uses the classes `Meter_Reader`, `StorageMgr`, `CommunicationMgr`,.
- Class `CommunicationMgr` implements the communication stack functionality allowing data to be sent and received through the channel. In order to enable a secure link, `CommunicationMgr` uses the `AuthenticationMgr` and `SecurityMgr` classes to setup a secure communication channel.
- Class `SecurityMgr` is used by `CommunicationMgr` class in order to encrypt/decrypt the outgoing/incoming messages. It also owns a table with the encryption keys of each RMR related by the RMR ID.

[2] SkipJack is the secret key encryption algorithm used by the US government in the Clipper chip and Fortezza PC card. It was implemented in tamper-resistant hardware and its structure had been classified since its introduction in 1993. More info at: http://www.cs.technion.ac.il/~biham/Reports/SkipJack/note1.html

- Class `AuthenticationMgr` is used by `CommunicationMgr` class in order to add/check the outgoing/incoming messages the correct MAC. It also is in possession of a table with the MAC keys of each RMR related by the RMR ID.
- Class `StorageMgr` is used by `SystemMgr` class and provides storage functionalities. It implements the operations needed to store and retrieve data to/from a local storage device.
- Class `Meter_Reader` is mainly a class for enabling RMR instances for the DC, where he can have a model of the currently under his "supervision" RMRs.

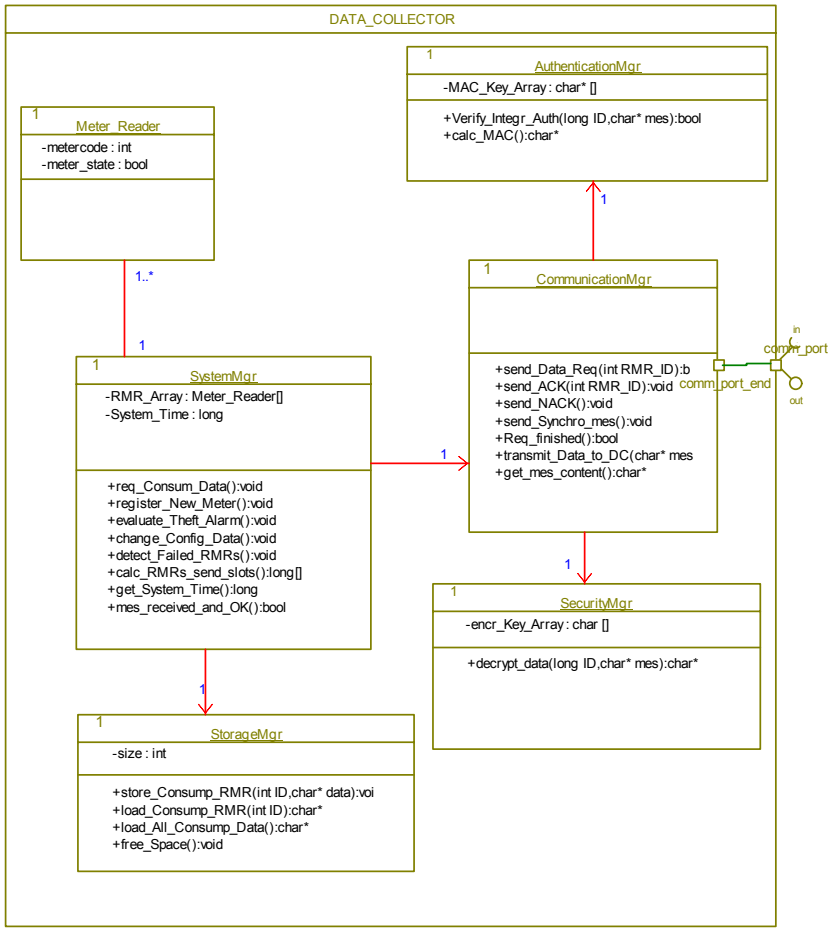

Fig. 2. Object Model diagram of the Data Collector

4.4.2 RMR

The object model diagram of the RMR can be seen in Figure 3 and its classes are similar to the ones of the data collector, of course enabling the other side of the connection. Here, the different possible sensors of the RMR are summarised in the class AllSensors, which is implemented only exemplary, as only one operation is needed for understanding the DC⇔protocol concepts. In a complete system model, this class has to be more detailed. This model is composed by the following classes:

- Class `SystemMgr` represents the overall controller in the RMR, and it operates over the other classes, e.g. receiving and actualising the synchronisation information, responding to the DC requests, storing the consumption data from the OCR sensor, enabling registering functionality for the first communication in a new network, etc. In order to do this, it uses the classes `AllSensors`, `StorageMgr`, `CommunicationMgr`,.
- Class `CommunicationMgr` implements the communication stack functionality allowing data to be sent and received on the channel. In order to enable a secure link, `CommunicationMgr` uses the `AuthenticationMgr` and `SecurityMgr` classes to setup a secure communication channel.
- Class `SecurityMgr` is used by `CommunicationMgr` class in order to encrypt/decrypt the outgoing/incoming messages. It also is in possession the specific encryption key of the RMR.
- Class `AuthenticationMgr` is used by `CommunicationMgr` class in order to add/check the outgoing/incoming messages the correct MAC. It also is in possession the specific MAC key of the RMR.
- Class `StorageMgr` is used by `SystemMgr` class and provides storage functionalities. It implements the operations needed to store and retrieve data to/from a local storage device.
- Class `AllSensors` provides all the values of the sensors, like battery level, the values of the analogue meter display or values of sensors that check for theft attempts.

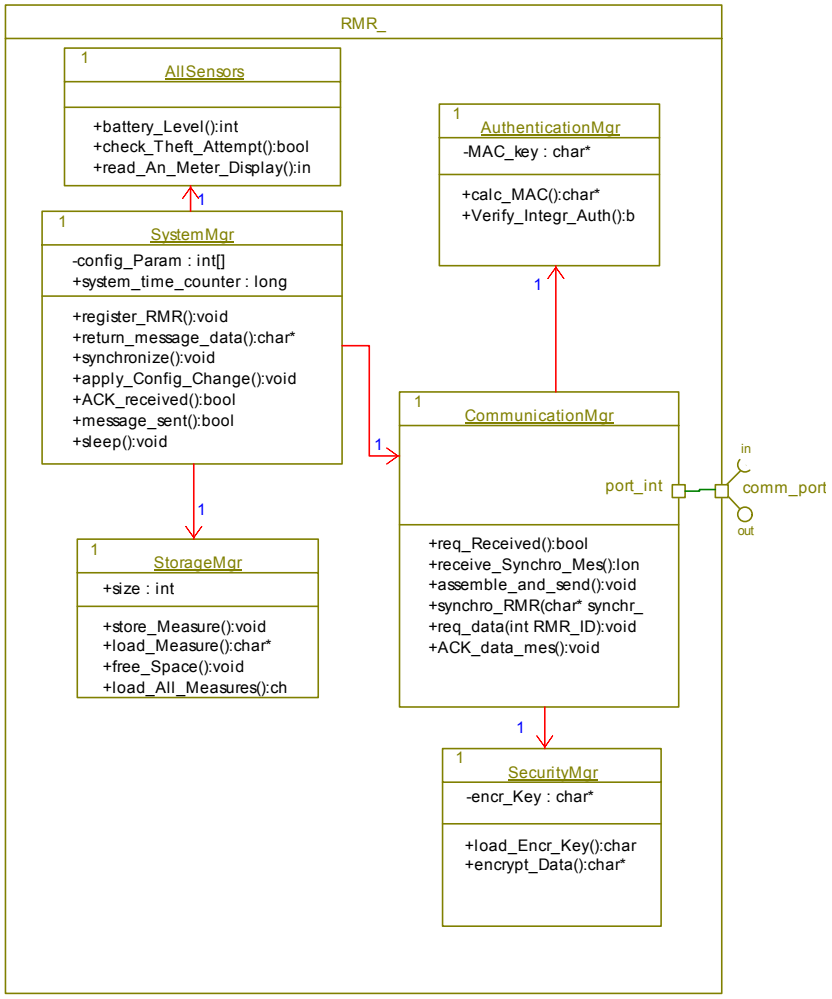

Fig. 3. Object model diagram of the RMR

4.5 UML Sequence diagrams

4.5.1 Data Collector

In Figure 4, a UML sequence diagram is provided, to show a typical example of a DC⇔RMR communication with the above explained object model diagram. In this case, only the DC side is considered. In Figure 5 we will talk about the RMR side

diagram. At the beginning, the DC SystemMgr connects to the CommunicationMgr to send a synchronisation message to all RMR in scope. The CommunicationMgr then requests the system time and a calculated time slot list from the DC, to provide the RMRs with an actual time stamp and with the information of when they will be requested for data transmission, to give the RMR the possibility to sleep. Then, the CommunicationMgr request the AuthenticationMgr to add a MAC to the message to send the message finally to the RMR CommunicationMgr instances. Here encryption is not necessary, as synchronisation is no confidential data. If required, encryption could be added. After the synchronisation phase, the RMR starts with a polling phase, were it request sequentially the consumption data from all the RMRs in its scope. This function calls an individual Data Request for each RMR, which is a function provided also by CommunicationMgr. After this request is sent to the RMR CommunicationMgr, the DC SystemMgr and the CommunicationMgr will be waiting for the RMR answer. If this answer arrives, CommunicationMgr instructs the SecurityMgr to decrypt the message, providing him the message and the also necessary RMR ID. Then the AuthenticationMgr is requested to verify the MAC on the message. If all the procedure has been correct, the SystemMgr gets a message that the data has arrived correctly. After this, the SystemMgr will request the decrypted data from the CommunicationMgr and order the CommunicationMgr, if the data is sinful, to send an ACK to the RMR instance. Here, the CommunicationMgr will have to wait some time if a resend of the message comes, as this would be the behaviour of the RMR if it would not get the DC ACK. If this doesn't happen, CommunicationMgr communicates to SystemMgr that the request process for this RMR has been accomplished in the right way. SystemMgr then will request that StorageMgr stores the corresponding consumption data. After this, the next request process for the next meter will be started.

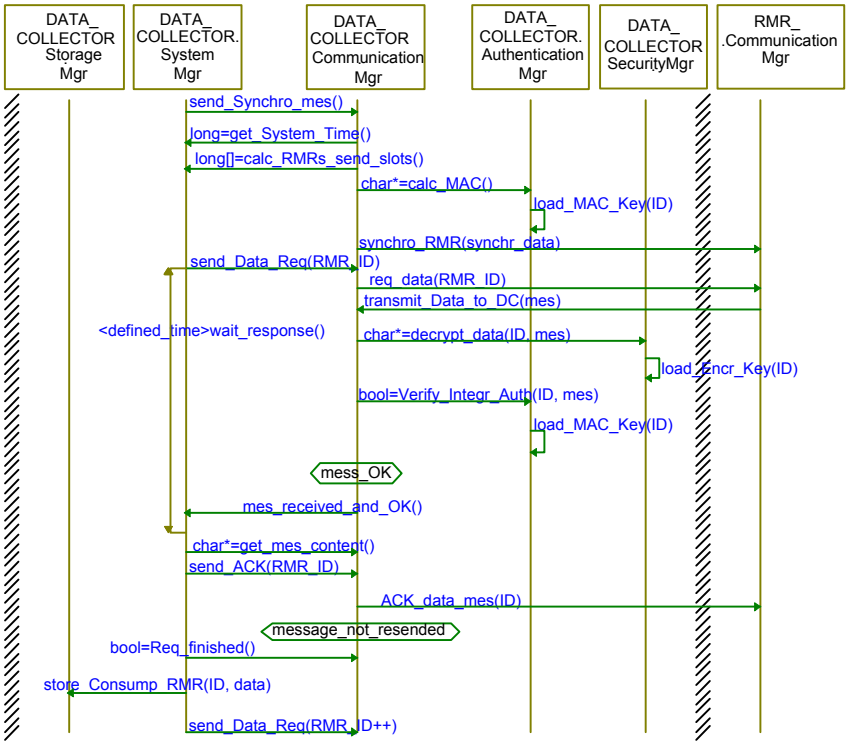

Fig. 4. Sequence diagram for a Data Collector instance

4.5.2 RMR

In Figure 5, a sequence diagram is provided, to show a typical example of a DC⇔RMR communication with the above explained object model diagram. In this case, only the RMR party is considered, excepting the necessary communication instance of the DC. At the start, when a synchronisation message is received from the DC, the message authentication is checked. After this, the `SystemMgr` updates the intern system time and depending of the received time slot where this RMR will be connected, RMR will be able to sleep for this time. Then, he will awake, and wait for the expected request. When the request from DC arrives, `SystemMgr` will first request from the `StorageMgr` all the in the last time stored analogue meter data. After this, the `SystemMgr` will calculate the IV for the packet, and then he will order `CommunicationMgr` to assemble and send the package. `CommunicationMgr`

will then first get the data and the IV from the `SystemMgr`, then he will request the `AuthenticationMgr` to calculate the MAC for the packet, and the `SecurityMgr` to encrypt the data payload. If all is done, he will assemble the message and send it. He will also communicate to the `SystemMgr`, that the message has been sent, and later, if an ACK for the package has arrived. If the ACK doesn't arrive in a certain time, the `SystemMgr` would order a second resend of the data. After the ACK, the request response phase is finished and the RMR can go into sleep modus again.

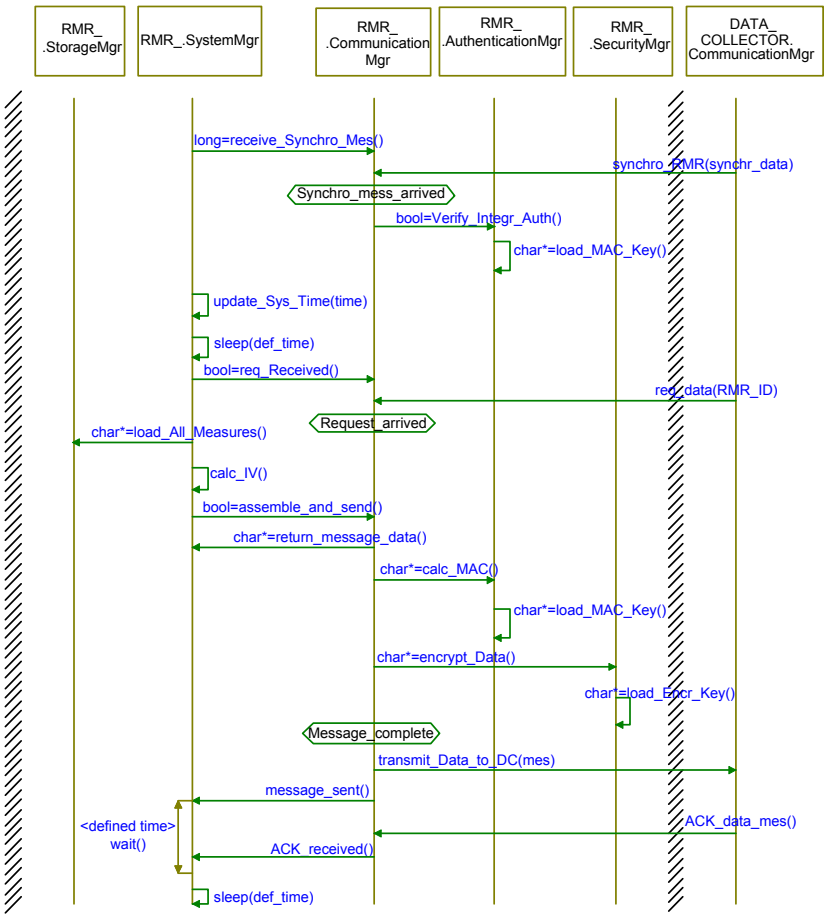

Fig. 5. Sequence diagram for an RMR instance

5 Conclusion and future work

After having analysed the state of the art for secure protocols that fit the centralised infrastructure network of our case study, which has low power constraints but at the same time doesn't require a too high security level, we decided to base our protocol design on the TinySec link-layer used in the Wireless Sensor Network area. In our opinion TinySec is the most mature security mechanism in the academic area. TinySec implies to use a CBC-MAC for authentication and if required, encrypt the data with the Skipjack encryption algorithm. The keying mechanism should be a per-link keying (for the MAC and encryption respectively), with the keys "burned" into the RMR at the production process, allowing these keying mechanism as the best security compromise. We propose a possible design implementation of our TinySec-based protocol by means of UML object model and sequence diagrams. This solution is built on top of TinyOS which is the operating system for low-power devices like e.g. sensor networks. The target platform for this solution could be an ASIC including a CPU, a memory and a RF transceiver. As described in [11] TinySec runs on platforms using Atmel (ATMega128L) which is the processor used in Motes processors radio platforms of Crossbow Technologies [19], company delivering products for sensor networks. Other people succeeded in porting TinySec on Texas Instrument microprocessors. The implementation of TinySec requires 256 bytes of RAM and 8152 bytes of ROM. The RF transceivers usually used are the RFM TR1000 radio and the Chipcon CC1000 radio which are the very popular in sensor networks devices and thus low-power devices. Given the broad range of platforms that TinySec runs on, according to [11] it seems that TinySec should be easily portable to both new processors and radio architectures.

As future work we think that our protocol should be verified by means of Model Checking [20] tools like SPIN [21] or verification tools like SATABS [22].

Another future works could be the porting of TinySec on a RF transceiver such the ones from Semtech, Nordic and Zensys and compare the obtained results in term of power consumption with the results obtained by running TinySec on RFM TR1000 radio and the Chipcon CC1000 radio.

In general in our future work, we can consider an implementation of the secure and low power protocol approach proposed in this paper, using a simulation environment. An adaptation of the TinySec link-layer in the sense of our case study could be also a good continuation. The results of these implementations should be then evaluated, to see, if the theoretical modelling done in these paper really is applicable to the real environment of an RMR system. Another future work could be to make a power analysis of the protocol proposed in this paper by means of StateC [16].

6 Acknowledgments

We would like to thank Mr. Claude Wieland from Xemtec for his appreciated feedback on this paper.

7 References

[1] Xemtec AG, [Online]: www.xemtec.com

[2] Sinem, Coleri Ergen, "ZigBee/IEEE 802.15.4 Summary", [Online]. Available: www.eecs.berkeley.edu/~csinem/ academic/publications/zigbee.pdf

[3] Kinney, Patrick, "ZigBee Technology: Wireless Control that Simply Works", October 2003, [Online]. Available: http://hometoys.com/htinews/oct03/articles/kinney/zigbee.htm

[4] Mayné, Jordi, "IEEE 802.15.4 y ZigBee", [Online]. Available: www.bairesrobotics.com.ar/data/IEEE_ZIGBEE_SILICA.pdf

[5] Krikke, Jan, "T-Engine: Japan's Ubiquitous Computing Architecture Is Ready for Prime Time", IEEE Pervasive Computing, April-June 2005 (Vol. 4, No. 2), [Online]. Available, http://csdl.computer.org/comp/mags/pc/2005/02/b2004.pdf

[6] [Online]: www.bluethoot.com

[7] [Online]: www.zigbee.org

[8] Montgomery, Steve, „Wi.232DTS vs. Zigbee; Comparing proprietary and standards based solutions", October 2004 [Online]. Available: http://www.radiotronix.com/datasheets/Wi232_vs_Zigbee.pdf

[9] SPINS project: [Online]. Available: http://www.ece.cmu.edu/~adrian/projects/mc2001/mc2001.pdf

[10] A. Perrig, R. Szewczyk, V.Wen, D. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *The 7th Annual International Conference on Mobile Computing and Networking, Rome, Italy*, pages 189–199, 2001

[11] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, Nov. 2004

[12] [Online]. Available: http://www.tinyos.net/

[13] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen. A survey of energy efficient network protocols for wireless networks. *Wireless Networks*, 7(4):343–358, 2001

[14] Gunnar Gaubatz, Jens-Peter Kaps, Erdinc Öztürk, Berk Sunar, State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks, 2005**,** [Online]. Available: www.crypto.wpi.edu/Publications/ Documents/GaubatzKapsPerSec05.pdf

[15] Y. Law, S. Etalle, and P. Hartel, "Assessing security in energy-efficient sensor networks," in *18th IFIP TC11 International Conference on Information Security, Security and Privacy in the Age of Uncertainty (SEC 2003)*, D. Gritzalis, S. D. C. di Vimercati, P. Samarati, and S. Katsikas, Eds. Kluwer Academic Publishers, May 2003, pp. 459–463. [Online]. Available: http://wwwhome.cs.utwente.nl/_ywlaw/pub/law03assessing.pdf

[16] L. Negri, A. Chiarini, "StateC: A power modelling and simulation flow for communication protocols", in *FDL'05 Proceedings, pp 555-566, Lausanne, Switzerland, September 27-30, 2005*

[17] Ronald Watro, Derrick Kong, Sue-fen Cuti, Charles Gardiner, Charles Lynn1 and Peter Kruus, "TinyPK: Securing Sensor Networks with Public Key Technology", *SASN'04*, October 25, 2004 Washington DC, web: portal.acm.org/ft_gateway.cfm?id=1029113&type=pdf

[18] [Online]. Available: www.uml.org

[19] [Online]. Available: www.xbow.com

[20] E. Clarke, O. Grumberg and D. Peled. *Model Checking.* MIT Press, 1999

[21] [Online]. Available: http://spinroot.com/spin/whatispin.html

[22] [Online]. Available: http://www.inf.ethz.ch/personal/daniekro/satabs/