Efficient Key Generation and Distribution on Wireless Sensor Networks

VÍCTOR ARIÑO PÉREZ

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden 2013

XR-EE-LCN 2013:011

Efficient Key Generation and Distribution on Wireless Sensor Networks

Author:
Víctor Ariño Pérez

Supervisor:
Panagiotis Papadimitratos

February 2013

Acknowledgements

Després de tot aquest temps, he arribat al final d'aquest camí. Ha estat llarg però ha estat un plaer de caminar al costat de tots aquells que hi he anat trobant.

Gràcies a la meva família. Mama i Xixa gràcies per tot el suport i afecte, per insistir en la meva educació i animar-me durant tots aquests anys. Sense vosaltres dubto que avui en dia fos qui sóc.

Als Batuts, per tants Bangs, tantes hores al poli i al bunquer, tants riures, previs i pizzes. Realment vau fer el temps a la UPC molt més agradable. Al Jordi i al Jaume per tots "l'aire no és teu" i "no pots". Al Roger, al Sergi i a l'Arnau perquè no sé què hagués fet els primers anys sense vosaltres i en què hauria perdut el temps sense Xplosió ni Distorsió.

I must thank you Panos, because you offered to be my examiner with just an interview, guided me through the process and accepted my strange schedules.

Thingsquare, Roger and Adam, thank you for the interview and the recommendation to Tado without which this work would not be possible.

Thanks to everyone in Tado and especially to Valentin, Jürgen and Tobias for teaching me so much in so few time about this crazy little things.

To all the people I met during last year in Stockholm. You made that 2012 in Stockholm the best year of my life. Luis, José, Aless, Esther and Adrien, thanks for all the parties, dinners, boats, mafias, all the experiences together and your friendship, which made it an unbeatable experience. Gracias, grazie, merci for making Ärvingevägen the coolest place in the whole city.

Jag måste tacka Google Translate också för sin hjälp i svenska kurser och för den fösta översättning av Sammanfattningen men en största tack till Mia för hennes finjustering.

Last but not least, I want to thank Julia for reading this before everyone else and make it readable for the rest, for insisting me to review it always one more time, for convincing me to come to Munich, being there every day to understand and support me and for so many other little things. Thank you.

Però el camí continua. De fet, sembla que això és una cruïlla i ara cal triar una nova direcció. Així doncs, seguiré aquesta fletxa que diu Munich i ja veurem al proper encreuament cap on anem. Endavant!

Abstract

Wireless Sensor Networks have become popular during the last years. The introduction of IPv6 which broadened the address space available, IEEE802.15.4 and adaption layers such as 6loWPAN have allowed the intercommunication of small devices. These networks are useful in many scenarios such as civil monitoring, mining, battlefield operations, as well as consumer products. Hence, practical security solutions for the intercommunication must be provided, ensuring privacy, authenticity, integrity and data freshness. In most cases, WSN nodes are not tamper-proof and have very limited available resources and capabilities which makes PKI currently not attractive for this environment. At the same time, key pre-distribution provide too low security for most applications. Therefore, the communication bootstrapping or the *key generation and distribution problem* is an important concern to be addressed with the additional difficulty of the constrained capabilities of WSN nodes. In this thesis, a solution to this problem is described. It makes use of ECDH and the curve K-163 for key exchange, AES-CCM-128 for symmetric encryption to lower the processing overhead and a partial challenge solving chain as well as a TAS to provide strong authentication. Several hash functions have been analysed as well as several random number generating approaches. At the same time, in order to fit the key generation and distribution algorithms together with the regular sensor operation, code optimizations were carried out on the cryptographic library Relic-Toolkit, reducing the memory footprint in 4KB; code reductions on Contiki OS allowed it to run using only 18KB of flash; and the peripheral drivers developed for the CC430 reduced as well the computation time. The solution allows to generate and distribute the keys in situ and is proved to be resilient to most adversaries while taking into account scalability, portability, energy consumption and making it suitable for consumer applications.

Keywords: Wireless Sensor Network, Pairing, Elliptic Curve Diffie-Hellman, Security, Key exchange

Sammanfattning

Trådlösa sensornätverk har blivit populära under de senaste åren. Införandet av IPv6, som breddade det tillgängliga adress utrymmet, IEEE802.15.4 och adaptionslager som 6lowpan har tillåtit förbindelsetrappor av små enheter. Dessa nätverk är användbara i många situationer såsom civilövervakning, gruvdrift och på slagfältet, de är även intressanta för konsumentprodukter. Därför måste praktiska säkerhetslösningar för kommunikation tillhandahållas, säkerställa sekretess, äkthet, integritet och datafärskhet. Ofta kan WSN noder inte manipuleras. Dessa noder har mycket begränsade resurser och kompetenser som gör PKI inte är attraktiv nuförtiden för denna miljö. Å andra sidan, ger för-fördelning en alltför låg säkerhet för de flesta tillämpningar. Därför är kommunikationen bootstra eller *nyckel produktion och distribution problemet* viktigt att fokusera på med den extra svårigheten som den begränsade kapaciteten på WSN noder ger. I denna avhandling har en lösning på detta problem beskrivits. Det använder sig av ECDH och kurvan K-163 för nyckelutbyte, AES-CCM-128 används för symmetrisk kryptering för att sänka bearbetnings overhead och en partial challenge solving chain utöver detta används TAS för att ge stark autentisering. Flera hash funktioner samt slumpmässiga metoder för nummergenerering har analyserats. Samtidigt, för att passa den viktigaste generationen och algoritmerdistributionen tillsammans med den ordinarie sensorfunktionen, har kodoptimeringar utförts på kryptografiska biblioteket Relic-Toolkit, som har reducerats 4KB; och Contiki OS källkod, gör det möjligt att köra i 18KB flash; och drivrutinerna har utvecklats för CC430 mikroprocessorn. Lösningen gör det möjligt att generera och drivrutinera nycklarna på plats och har visat sig kunna stå emot de flesta problemen samtidigt som hänsyn tas på skalbarhet, överförbarhet och energiförbrukning samt att göra den lämplig för konsumentprodukter.

Nyckelord: trådlösa sensornätverk, pairing, Elliptic Curve Diffie-Hellman, säkerhet, lösenord utdelning

Contents

List of Figures

List of Tables

List of Symbols

AES	Advanced Encryption Standard
BDHP	Bilinear Diffie Hellman Problem
BER	Bit Error Ratio
BIBD	Balanced Incomplete Block Design
BS	Base Station
CA	Certification Agency
CH	Cluster Head
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access
DH	Diffie-Hellman Key Exchange
DLP	Discrete Logarithm Problem
DoS	Denial of Service
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem
GW	Gateway
HIP	Host Identity Protocol
HMAC	Hash-based Message Authentication Code
HVAC	Heating, Ventilation and Air Conditioning
IBC	Identity Based Cryptography
IDS	Intrusion Detection System
IT	Information Technology

KDF	Key Derivation Function
KDS	Key Distribution Server
LHIP	Lightweight HIP
LLA	Link-local Address
LPM3	Low Power Mode 3
MAC	Media Access Control
MASLKE	Mobile Agent Secure Location Key Establishment
MIC	Message Integrity Code
MiM	Man-in-the-Middle
NIST	National Institute of Standards and Technology
nonce	number used only once
NS	Neighbour Solicitation
OOB	Out of Band
P2P	Peer-to-Peer
PAN	Personal Area Network
PKC	Public Key Cryptography
PKI	Public key Infrastructure
PRNG	Pseudo Random Number Generator
RA	Route Advertisement
RDC	Radio Duty Cycle
RPL	Routing Protocol for Low-Power and Lossy Networks
RSA	Rivest, Shamir and Adleman
SICS	Swedish Institute of Computer Science
SKC	Symmetric Key Cryptography
SPINS	Security Protocols for Sensor Networks
TAS	Trusted Authentication Server
TBS	Trusted Base Station
TESLA	Time Efficient Stream Loss-tolerant Authentication
TRNG	True Random Number Generator
UDP	User Datagram Protocol
WSN	Wireless Sensor Network

Chapter 1

Introduction

A Wireless Sensor Network (WSN) is a distributed network of sensoring devices (usually called nodes or motes) that monitor physical phenomenons such as temperature, radiation or humidity and can be used for surveillance of an area which makes them really attractive for consumer applications and for battlefield usage.

A complete WSN is formed of a variable amount of nodes, it can be from few units to several thousands and sometimes the number is ever-changing: there is no fix number of nodes forming the network, as new nodes may join and other break and be replaced or just removed. The nodes are typically cheap, devices with few resources and therefore very constrained in several aspects. The growth of this kind of networks, partially enforced by the idea of *The Internet of Things*, raises the problem of securing the communication of those networks within these very particular conditions. The Internet of Things roots on the thought of all sort of objects connected to the Internet with an own identification, being able to share data and interact with each other.

Nodes must be able to set up link-wise and end-to-end security while these security procedures shall not affect their regular performance. A solution to solve the secure communication bootstrap issue is proposed in this master thesis. In the Section Background, an overview of the particularities of these networks is given and the problem and motivation are stated in Section 1.2. Finally in Thesis Organization an outline of the further sections is given.

1.1 Background

The growth of WSNs together with *The Internet of Things* and technologies like IPv6, 6loWPAN and IEEE802.15.4 has opened the door once more to research on key generation and distribution schemes. The importance of this field in WSN, is key in the future success of these networks.

A typical compiled OpenSSL cryptographic library has a size of around 620KB, which makes it not feasible to embed it into a 30-256KB microcontroller. Additionally, the microcontrollers for sensors nodes are usually powered by batteries and operate at a frequency range between 4 and 50 MHz in comparison to the 3GHz microprocessors of desktop computers. Hence, operations take longer time and intensive computing can easily led to battery exhaustion. Because of all this, the implementation of security standards such as Public key Infrastructure (PKI) is a challenge and not always feasible. Although research

focused during the last decade on more insecure and simple key pre-distribution schemes, new studies demonstrate that optimized Public Key Cryptography (PKC) is affordable for WSNs.

First, the specific aspects of WSN are discussed in 1.1.1, and then an overview on information security and key exchange is given in 1.1.2.

1.1.1 Wireless Sensor Networks

As already mentioned, the nodes forming a WSN are usually inexpensive and not tamper-proof. This suggests that the nodes have very limited computational resources and memory and they can be attacked physically. Most frequently, the nodes are battery powered which adds another constraint to any kind of energy consuming operation. Although they can be arranged in different topologies, the most common one and easiest to protect the hierarchical topology[53]. In such a topology the central node, known as Cluster Head (CH) is usually more powerful than the rest, as it has to manage many data inputs. Another possible topology is Peer-to-Peer (P2P) where all the nodes have the same role within the network and there is no node more important than the rest. Of course, variants of both are also possible.

Typically the CH is connected to a Gateway (GW) node which is the door between the network and the Internet. The role of a GW is the same as of a router, it can hold some access control mechanisms but mainly it acts as a network converter, e.g. from IEEE802.15.4 to Ethernet. A hierarchical WSN topology is depicted in Figure 1.1. Normally, WSN are divided into clusters for better supervision, the CH is the responsible for an area and report to the Base Station (BS) (or Data Sink), in smaller networks, the CH assumes the role of BS.

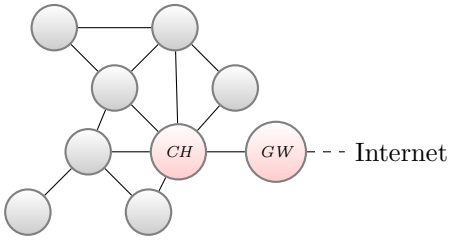


Figure 1.1: *Hierarchical Wireless Sensor Network topology*

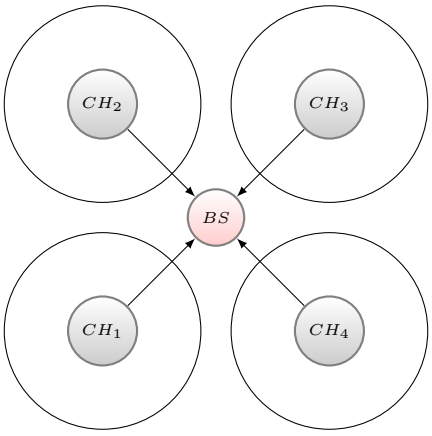


Figure 1.2: *Clustered Wireless Sensor Network*

The nodes take data samples of environment phenomenons and send them to the data sink, receive orders from the CH or operate depending on readings of other nodes. For instance,

in a parking lot, nodes indicating free places will turn on and off green and red LEDs based on the availability of a parking spot in the nearby. At the same time they will send status information to a main node which updates a display showing the number of free places in each storey. In a more extreme situation where nodes are in the deep corridors of a mine, hence in places difficult to access, they report air oxygen levels to a main controller, which will trigger harder ventilation when required in order to protect miners.

However, not all the interconnected nodes must be of the same nature, a toaster equipped with IPv6 connectivity for alerting of ready toasts may be used at some point to forward packets of the house lighting system as well. This cooperative idea is the main goal of *The Internet of Things* and the use of standards communication protocols make it possible.

Despite of the benefits that a WSN presents, the increase of their use unveil new threats and bring opportunities for attackers. Therefore, security is mandatory for many applications and must be provided.

1.1.2 Information Security

Commonly, information security and cryptography are confused with *confidentiality* and encryption, which ensures that the message can just be read by the genuine consignee. But this is neither the only security aim nor the most important one when speaking of information security. Other goals such as *integrity*, which proves that the message has not been altered from the source; *availability*, which makes sure that the data is available whenever needed; *authenticity*, which confirms that the message comes from where claimed; or *data freshness*, which will allow the receiver to detect when a new message has arrived are sometimes more important. However, these purposes are difficult to provide in an autonomous WSN without trusted third parties and previously unknown node placement.

In order to ensure the previous goals, some secret information between the communicating parties must be shared. Depending on the way this secret information is managed, two tendencies are identified: *Symmetric Key Cryptography* and *Public Key Cryptography*.

In Symmetric Key Cryptography, security bases in a shared key, only known by the communicating ends. This key must be transmitted in a secure way, usually using a different communication channel. Public Key Cryptography refers to a type of cryptography which uses asymmetric algorithms, based on the use of a key pair consisting in two keys: a private one and a public one. Therefore, when one of the keys is used to cipher an information, only the other key of the pair can be used to decrypt it. It is useful for signing and guarantees most of the previously mentioned security goals. These algorithms base their strength on mathematical problems with difficult solution such as the Discrete Logarithm Problem (DLP) or big number factorizations used by Rivest, Shamir and Adleman (RSA).

The level of understanding of the methods used also reveals four levels of security: *Security by Obscurity*, *Presumed Security*, *Computational Security* and *Provable Security*, being *Computational Security* the most widely used in modern Information Technology (IT) systems.

Security by Obscurity is a term used to describe the notion of security achieved by keeping the algorithms and mechanisms secret following the idea that ignorance will make it impossible to find a security breach. *Presumed Security* is the sort of security which is considered secure because no one has (yet) been able to prove the opposite.

In *Computational Security*, it is considered that a system is secure as high computational

power, hence time, is required in order to break the security. This makes it most of the times not worthy to attempt to crack it. This is the case of for instance RSA and mostly all modern security schemes, such as DLP and Elliptic Curve Discrete Logarithm Problem (ECDLP).

Provable Security is the most rare case of security as security can be proved. An example of it is *Semantic Encryption* which makes a different cryptogram every time for the same cleartext and password inputs. Nevertheless, this scheme is rarely used in favour of *Computational Security* methods.

1.2 Problem Statement

In any security context, additional data have to be transmitted appended to the original message, in order to achieve some of the security goals (e.g. authenticity and integrity) described in the previous section. This turns into additional communication overhead, which is present in all the secured communications. Furthermore, as can be deduced, encryption operations or key derivation algorithms require additional (usually intensive) processing time, hence producing processing overhead. Both communication and processing overheads lead to a lower battery life, which is a precious resource for WSN. It is clear that a compromise between security and battery life must be achieved.

The widely researched pre-distributed keys approach, consisting in storing keys into the nodes in provisioning time, requires few overhead in the communication and almost no processing overhead since the keys are already pre-loaded in the nodes. However, since the unawareness of the final location of a sensor is a common situation for mass produced devices, key pre-distribution is not always the best option: the number of keys to be preloaded can be huge without deployment knowledge. Additionally, it does not scale and implies high risks for the entire network when a node is compromised as it may disclose all available keys. Other authors considered key derivation from a master secret or a polynomial. While it requires low or no communication overhead, it may allow an attacker to compute all the keys within the network if the secret or the polynomial is known, which is easy since the nodes are not tamper-proof. Finally, the traditional use of PKC and PKI is discarded for the high computational requirements, opting for Elliptic Curve Cryptography (ECC) which is a less processor abusive alternative. Albeit requiring less dedicated processor time, an ECC signature on a MSP430 microcontroller still takes over $1s$[47] –in contrast to the $12s$ required by a signature using RSA–. Furthermore, the big memory overhead of those schemes, due to the code footprint of cryptographic libraries, is also an important constraint to consider. All this, together with the larger communication overhead compared to the previous approaches, make the application of traditional PKC using ECC not interesting for most of the sensing systems in favor of less secure approaches.

Thus, the main issue lies in the secure communication bootstrap: the way keys are generated and distributed among the sensors. The most secure approach makes the sensors act autonomously and generate their own keys from information obtained using algorithms such as Diffie-Hellman Key Exchange (DH), but their high processing requirements makes them not suitable. Pre-distribution schemes do not scale well nor protect efficiently the nodes in case of tampering as well as polynomial or key derivation approaches, which also have proved to be insecure. However, their memory, processing and communication overhead is smaller. An efficient solution for in situ transparent key generation and distribution with a good trade-off between security and overhead and a good defense in case of node capture must

be provided. This thesis targets the key generation and distribution as an issue in a link and application layer levels[1].

1.2.1 Motivation

It can be seen in the Problem Statement (Section 1.2) and from the literature review (Chapter 2) that key generation and distribution in WSN is a difficult problem to solve. A generic solution which offers a good trade-off between the use of the resources and the security level achieved does not exist, hence must be provided.

The current approaches can be categorized mainly in three big groups: *Key pre-distribution*, *Polynomial derivation* and *PKC*. Other approaches are a combination of the previous or key derivation from a master key.

- Key pre-distribution has the advantage of no key computation and few or no exchange messages depending on the distribution scheme. It is widely described in the Section 2.1.1. Its main drawback is that once a key is disclosed, all the links using that key are compromised and it is highly vulnerable to selective node tampering which can lead to a key exhaustion. Additionally, the memory overhead grows exponentially with the size of the network when probabilistic schemes are used without post-deployment knowledge.

- Polynomial approaches and master secret derivation offer a good arrangement between memory, computation and no communication overhead. These approaches allow to compute the communication keys from a polynomial or secret data, hence they have the inconvenience that once the polynomial is known, an external attacker can compute all the keys of all the links within the network.

- PKC has the advantage of providing high computational security, unpredictable and independent link keys, hence a compromised node only may compromise its links and eases the use authenticative signatures. The main drawback for this technology is the high computational requirements additionally to the memory overhead for the cryptographic libraries required.

There is no known method existing that offers a good trade-off between all of them, keeping low the memory, communication and computation overheads. Additionally, most of the researched methods do not contemplate consumer applications, which makes them neither usable nor error-proof for industry products. Furthermore, many proposed solutions are platform specific which makes them not portable to different platforms and breaks with the idea of the Internet of Things. A new solution has to be developed, which offers a good level of security, relatively small overheads and aims usability for consumer products keeping in mind its portability to different platforms.

1.2.2 Objectives

The objective of this thesis is to develop a key generation and distribution method which offers a good compromise between memory, computation and communication overheads while offering a high level of security for WSN communications.

[1] Physical communicative attacks such as *jamming* are out of its scope of this master thesis

The approach must allow in situ key generation, maintaining a different key per link and a group key for multicast messages. This must be energy-efficient, not cpu abusive and scalable according to the needs of the nature of a WSN. Furthermore, the solution has to be integrable in Tado GmbH network hence it must be developed in C and work as a transparent Contiki OS layer for the MSP430, CC430 and Cortex M3 processors, which are the current used microprocessors in Tado motes, and any future addition, therefore portable. It has to have a small memory footprint since it must co-exist with current firmware in the nodes and has to provide a good level of security for avoiding possible attacks and protect the customers as well as the company network from possible attacks and reverse engineering.

Expected Results

After literature review (see Chapter 2), it can be affirmed that the optimal way to achieve the goal of the thesis is by an efficient use of ECC. Hence, general purpose cryptographic libraries must be adapted for their use in embedded devices, in order to lower the energy consumption and memory overhead. It is expected that the use of Elliptic Curve Diffie-Hellman (ECDH) is affordable for secret information exchange allowing the parties to derive a link key while using symmetric encryption for the rest of the communication, lowering the number of exchanged messages and their size.

Another straightforward expectation from the bibliography is that any block ciphering algorithm can cause far too much overhead on the communication. Hence, AES-CCM or AES-CBC-stealing[51] will provide important savings in the communication, which is one of the most energy expensive components of a WSN[7].

Finally, after the inspection of the current state of Tado's devices and firmwares, deep software optimizations are required in order to fit the protocol to implement together with the cryptographic library and the regular device application.

1.3 Thesis Organization

This thesis report is organized in seven chapters. This one presents an overview on WSN, the key generation and distribution problem as well as the objectives and scope of the thesis. It is followed by a review of the solutions proposed by researchers. In the Environment chapter, an examination of the available standards is given as well as a description of the hardware available for testing. Chapter 4 offers a description of tools and algorithms in order to generate the keys on the nodes while Chapter 5 relates how keys should be exchanged between the different nodes in the network. In Chapter 6 a security and overhead analysis of the proposed solution is given and finally, the conclusions are stated in Chapter 7.

Chapter 2

Related Work

The key generation and distribution problem has been concern of researchers for a long time. A traditional solution to this problem consists in the use of some functions that are easy to compute in one way but very difficult to reverse such as the DLP. However, this is not always suitable for WSN[11, 37] because of the high computation required. Albeit some have managed to implement PKC in small nodes[26, 49], the computational time for encryption, decryption, signature generation and verification or even the memory overhead makes PKC not suitable for this sort of networks.

Research has shown that the transmission of 1 single byte of data in WSN is approximately as expensive as 800-1000 32-bit processor instructions[7]. Hence, it should be considered to make the key generation and distribution system less verbose in favour of computation. This must be the main goal of good schemes.

In this chapter, several solutions proposed by researchers are reviewed. The reviews are divided in five main categories depending on the nature of the solution, being these: *key pre-distribution*, *polynomial derivation*, *Public Key Cryptography*, *physical observations* and *hybrid* schemes.

2.1 Key Pre-Distribution

The pre-distribution schemes solve the key generation and distribution problem by pre-loading secret information in the nodes before deployment, different types be distinguished depending on the nature of the pre-loaded information. These approaches have no processing overhead, since the keys are already pre-loaded and ready to use, while offering very few communication overhead. However, they have some important issues as discussed in the sections below.

2.1.1 Single key pre-distribution

The most straightforward solution to solve the key generation and distribution problem may be to pre-load the same shared key in all the node, which is used to secure the communication. However, a node capture would disclose the key and compromise the whole network. Hence, it is not a good approach for networks with tamperable nodes.

An attempt to solve the previous scenario is to use a different key per node forcing all the nodes to know all the keys of at least its neighbours, hence causing a huge memory overhead for large networks and preventing scalability. Still, a node capture may break as well neighbouring links since neighbouring keys are disclosed, leading to an unsecure network.

A second attempt to solve this problem in an upright way is the use of a trusted third party[12, 53, 54] which knows all the keys for different nodes, and to which the members of the network can request keys to communicate to other nodes. This direct solution has scalability problems as a request message from a node in an N^{th} level of a network tree will need $2(N-1)$ message forwards before receiving the key. Additionally, a captured node may disclose the keys of all its neighbours voiding them.

2.1.2 Master Key Derivation

This type of key pre-distribution bases its strength in the ability of the nodes to derive the keys from a master, pre-loaded key. A paper from Blom in 1985[9] proposes to provide nodes with enough information in a way that the cooperation of the nodes allow one of them to build a pairwise key. Despite it does not require big computation capabilities or memory overhead, the capture of the master key allows an attacker to compute all the keys for the nodes.

One of the most recognised and discussed master key-derivation approaches is Security Protocols for Sensor Networks (SPINS)[46]. All the nodes share a secret key with the trusted base station, from which the keying information of a link is derived. Semantic cryptography[1], data freshness, weak authentication and very low communication overhead are considered. It uses $\mu TESLA$, an efficient stream authentication scheme, symmetric encryption for authenticated broadcasts and relies on loose time synchronization. The schema is designed for very limited sensors but it presents some drawbacks for consumer and large scale applications: a) the required time synchronization is not always possible, b) the keys are stored in the gateway which may suffer from limited storage capabilities and c) only BS to node and vice versa communications are contemplated.

[12] proposes considering the gateway a Trusted Base Station (TBS). An online server to the network is added in order to solve the key storage problem and multiple authentication keys are pre-loaded in each node. The node requests authentication to the GW which queries the online server. The latter replies with a challenge for the node to solve. Once a node is authenticated, both parties (node and GW) can derive a session key K_c from the challenge. Despite the improvement it does not authenticate the network and therefore if the key derivation function is captured, the system is vulnerable to eavesdropping and Man-in-the-Middle (MiM) attacks.

Nehra and Patel[41] propose Mobile Agent Secure Location Key Establishment (MASLKE) in order to fulfil the leakages of the previous solutions. In MASLKE, the CH is in charge of key generation and distribution. It is considered that each node in a cluster is able to talk to its CH. At communication bootstrap, every node generates a nonce (x_i), encrypts it with the pre-shared individual key and broadcasts it (y_i). The CH selects N nonces (r_i), with N being the number of networks in the cluster, and encrypts every nonce with the pre-shared individual key of each of the nodes (t_i). All the encrypted messages are delivered to all the nodes individually. At this point, the nodes have all the encrypted messages broadcasted by others, their piece of decrypted information and the decrypted information they received from the CH. The individual key is generated as the hash of all the encrypted keys together with the two nonces that the node has been able to decrypt. In Equation 2.1, key generation for a

[1]Two instances of the same message result in two different cryptograms. This is achieved, for instance, appending random information to the original cleartext.

certain node is transcribed, where g is the generator of a cyclic group G, while in Equation 2.2 the way in which the keys are generated is described.

$$CH: \quad t_i = E_{p_i}(g^{r_i}) \qquad N: \quad y_i = E_{p_i}(g^{x_i}) \tag{2.1}$$

$$K_i = H_1(y_1, y_2, \ldots, y_n, g^{x_i r_i}) \tag{2.2}$$

A captured node may not disclose more information than the relative to itself. Since all the nodes must send one message the communication overhead is very low as well as the computational requirements. However, the assumption that all nodes can listen to all broadcasts may not be realistic and a tampered CH can compromise the entire cluster as it holds all the keys. Nevertheless, this solution has almost no communication overhead and requires few memory and computation imprint.

Tufail et al.[57] propose to pre-load nodes with a hashed version of a master key K_m together with the node ID, $K_I = H(K_m \mid\mid ID)$. When a node joins the network, it sends this K_I, the node ID and a temporary generated session key K_S to the gateway, encrypted with the gateway public key (K_g), which was previously received of a Route Advertisement (RA) message. Then the gateway replies with a group key encrypted with K_S and the node erases the two previous keys (K_S and K_m). An important drawback is that this approach does not authenticate the gateway and forces the nodes to be capable to perform public key encryption, which is sometimes not feasible[15, 37]. Since the network uses one single group key to cipher the communications, a captured node compromises the entire network. Additionally, no key renewal is possible since the information is deleted from the node after deployment.

Others[64] recommend the use of three different keys, two of those are preloaded and the third is computed from the previous two and random data. The preloaded keys are network, sensor and cluster specific. The message forwarding in this structure causes huge overhead as every level adds its own Message Integrity Code (MIC) and encryption parameters. However, it is a good approach to separate the environments using different keys. A wormhole attack may induce a packet to loop indefinitely making it grow boundlessly. All the information of the network can be read out when the network key is captured as only sensor-CH information is protected individually. Hence, the capture of a node discloses all shared keys.

2.1.3 Key Pool Pre-loading

Eschenauer and Gligor[15] proposed one of the first key pool pre-loading schemes. A TBS, which can be an external computer, generates a key pool of K keys, with $K > N$, being N is the number of nodes. Every node has an aleatory sub-set of $M \mid M < N < K$ keys pre-loaded before deployment. During neighbour discovery phase, a node broadcasts all the IDs of the keys it holds to all its neighbors. This way, two neighbor nodes with one or more matching keys can agree in using a certain one as link key. Despite the huge memory prerequisites for large networks, the proposal requires no computation overhead and almost no communication overhead.

Nevertheless, there is no guarantee that two neighboring nodes share a key. Hence, a huge key pool must be pre-loaded into the nodes to ensure a certain level of probability of key sharing; and when a node is compromised, revocation of all the held keys is mandatory, breaking several links and allowing an *Smart attacker* (see Section 6.1.1) to exhaust all the

key pool easily. All of this is concern of Chang et al.[11] who presented the *q-composite* key pre-loading. The main idea behind this is that two nodes must share at least q keys to compute the pairwise key. Additionally, the size of the initial key-pool ($|S|$) is calculated given the constraint of the probability that two nodes share at least q keys and the number of keys that a node can hold (m). The probability is as given in Equation (2.4).

$$p(i) = \frac{\binom{|S|}{i}\binom{|S|-i}{2(m-i)}\binom{2(m-i)}{m-i}}{\binom{|S|}{m}^2} \tag{2.3}$$

$$p_{connect} = 1 - (p(0) + p(1) + \ldots + p(q-1)) \tag{2.4}$$

It can be seen from (2.4) that the probability increases when the pool size $|S|$ decreases, making it not suitable for very small networks. Although [12] pointed out that the connection probability ($p_{connect}$) may be increased by previous knowledge of final distribution of nodes in the field, this is not always possible.

Chang et al.[11] also propose *multi-path reinforcement* to improve resilience of the network by sacrificing resources. Additionally to a *random pairwise scheme* to make sure that if a node A shares exactly the same K keys with the nodes B and C, both neighbors of A can ensure that the received message comes from B or from C, thus weak authentication. *Multi-path reinforcement* requires the existence of disjoint paths which may not always be the case, and the *random pairwise scheme* does not scale as mentioned in [13, 14].

At the same time, Di Pietro et al.[14] suggest pseudo-random key pre-loading for a more efficient key discovery procedure. A node knows how to compute the keys it shares with another node. While in [11], all shared keys XORed to compute the communication key, a cooperative scheme is advised. When a node A wants to communicate with node B, it first asks a set of nodes ($N \in \zeta = N_1, \ldots, N_m$) for a hashed ID_A together with their $K_{N_i,B}$ as in Equation (2.5) where H stands for the hashing function. The pairwise key between A and B is the XOR of the shared keys between A and B XORed with all the received hashes. A tells B the resultant key and the nodes in ζ, thus B computes the key as well.

$$k^{\zeta}_{A,B} = k_{A,B} \oplus (\bigoplus_{c \in \zeta} H(ID_A \mid\mid K_{N_i,B})) \tag{2.5}$$

The resultant key does not depend only on the keys of the node. However, the capture of one node may offer the seed and the algorithm used to compute the IDs of the keys a certain node has. Since A will send B the set of nodes that participated in the key creation, an *smart attacker* may be aware of which nodes to capture. As an attacker might know which keys are shared with every node, it can trace a map of keying exhausting all the keys in the network and breaking all the links.

Another alternative is to use multiple key spaces[55]. For a lower number of keys pre-loaded, key sharing probability of 1 can be guaranteed. The approach bases in a Balanced Incomplete Block Design (BIBD) which defines an arrangement of n different objects into b in such a way that each block contains exactly k different objects.

Therefore, the final location of the nodes must be known in order to distribute the keys, making it not attractive for many scenarios neither for small sized networks, as many of the proposals described in this section.

The use of hash-chains is recommended in [8], hashing the pre-loaded keys $n \; mod \; L$ times

before storing them in the nodes where n is the ID of the node. Subsequently, the session key will be derived from all the common keys. Thus, every link key will be different even if based in the same base key. The node with the lower ID computes hashes x more times to compute the shared session key. This is claimed to enhance the resilience of the network up to 40%.

Combinatorial approaches for key pre-distribution are discussed in [56]. Two types of keys are considered: one for intra-region communication and another one for inter-region communication. The CHs have at most one common key, thus a compromised agent does not affect the other CHs and no key discovery algorithm has to be triggered. Their approach bases on dividing the network in regions small enough to fit all the neighboring keys of a certain region into that node's memory. During key discovery the approach only broadcasts node IDs so no key index is disclosed, which makes it very difficult for an smart attacker.

2.2 Polynomial

Shen and Chien[54] propose a symmetric bivariate polynomial to compute the keys (Equation (2.6)). Only with the node IDs a link key can be computed and hashed afterwards. The main issue in this scheme is that once a node is captured, the polynomial can be obtained and the keys of all links can be computed. In [53], two scenarios for the scheme above are described: *online generation* and *offline generation*.

$$K_{CH,S} = H(f_{CH-S_i}(S_i, CH_x)) \tag{2.6}$$

Online Generation: this is the case when a Key Distribution Server (KDS) is present in the network. The joining node (i) computes the key of the pairwise key with CH as the hash for the polynomial result using the node ID and the CH ID (x). Afterwards, the polynomial is removed. At the same time the CH, which does not know the polynomial, requests a new key to the KDS for the link with S_i. A captured node before deployment will provide the attacker with the polynomial to compute all link keys.

Offline Generation: this is the situation when a KDS is not present. Then the nodes must be preloaded with a known node-CH key ($K_{CH_x,N}$) and a CH-node key encrypted with the BS-CH key ($E_{K_{CH_x,BS}}(K_{CH_x,N})$). This way the CH will be able to decrypt the message. Even if it belongs to another cluster head, it will be forwarded and re-encrypted by the CHs network. In this approach the nodes final position does not affect their capability to join the network, though the fact that only node-CH communication is considered.

García-Monchón et al.[18] propose to use small digital certificates and a symmetric bivariate polynomial for key exchange using the certificates as node-ids for medical WSN.

In Min-Qing et al.[36] approach, every node generates a random secret r_i and sends $E_{K_{pub}}(H_i||ID_i)$ (Equation 2.7) to the base station using the base station public key (K_{pub}). The BS decrypts all received messages and generates a polynomial such as $p(x) = (x - H_1)\dots(x - H_n) mod\ p$, then sends $g(x) = (p(x) + K) mod\ p$ to all nodes, but only valid nodes can retrieve the key K. When the key needs to be updated, a new key is broadcasted in the same way. As every node stores very few data, there is not much communication overhead but all nodes use the same link key which is at the same time an advantage and a drawback. However, the base station must not be captured. The network does not authenticate the nodes, and any external node knowing the procedure may join if no upper layers of authentication

are introduced.

$$H_i = H(ID_i \mid\mid r_i) mod\ p \tag{2.7}$$

2.3 Public Key Cryptography

Steffen et al.[47] confirm the infeasibility of software implementation for PKC on WSN. Long computing cycles make energy consumption behave poorly. They also compare RSA with ECC on several processors and configurations showing that ECC is around 10 times more efficient for the same degree of security. Nevertheless, they propose a hardware accelerator to be used as a peripheral and proving that hardware accelerators can help to reduce the power consumption while providing as well a huge speed improvement when calculating PKC operations. Others consider that this is not necessary and the increase of speed can be achieved by pre-loading some already generated random data for the first pairing[63] basing the security on the Bilinear Diffie Hellman Problem (BDHP).

Some proposed to use Host Identity Protocol (HIP)[38, 39] for message authentication using PKI. HIP is used on top of IPv4 or IPv6 to identify hosts. It is "a self generated PKI certificate which is mean to be the host identity"[20]. This allows separation of the identifier and locator roles of the IP address. However, to be completely trusted and to allow authorization, the certificate must be signed by a trusted Certification Agency (CA), which adds a huge amount of overhead to the communication.

In [20], it is proposed Lightweight HIP (LHIP) and several alternatives to the original HIP protocol such as: to dispense DH, Digital Signature Algorithm (DSA) and to reduce the key size. They propose the use of Hash-based Message Authentication Codes (HMACs) for message signing and to refer to Time Efficient Stream Loss-tolerant Authentication (TESLA)[45] for time based signatures.

HIP is proposed to be the main network layer security builder[26] for WSN. It is meant to use Certificates signed by a trusted CA together with ECC to improve the bad performance observed in WSNs using traditional cryptographic methods such as RSA or DSA; and replacing DH by ECDH to derive keying material for link based Advanced Encryption Standard (AES) communication. Nevertheless, in [26] it is proposed to use *Binary Trees* for lightweight certificate test together with polynomial key exchange (see Section 2.2). It is claimed to improve dramatically the performance of PKC on WSN.

Some[25, 40] considered the use of Identity Based Cryptography (IBC) based on ECC and *bilinear pairing*. In those schemes the node ID (serial number, Link-local Address (LLA), etc.) is used as the public key. The energy consumption is highly reduced by this procedure thanks to the avoidance of the exchange of messages for key generation. Another interesting option presented is to use the IPv6 address of the node as a public key and generate the private keys of the node during pre-deployment by a trusted third party[40]. This results in a little less latency compared to the previous in spite of to higher energy consumption.

2.4 Physical Observations

Nitinawarat et al.[42] point out that it is possible to generate a key using the signal reception since a pair of terminals will observe correlated signals that are independent of all other pairs

of signals. In [60], it is proposed to use the phase of the received signal as it will be equivalent in both communication ways. This way, the key is generated by two pairs or more, each transmitting in a certain moment and computing the addition of the phase differences. The security of the schema remains in that it is very difficult to receive the same phase in another location different from the genuine nodes. However, time synchronization is fundamental and may not be possible in some platforms. Furthermore, it is difficult to detect when an additional signal has been added by an attacker and the accuracy to detect the phase of the received signal is critical, as the accumulation of estimation errors when generating a group key may cause the procedure fail. At the same time, the group size is limited by the required Bit Error Ratio (BER). It is a very sophisticated method which requires low level signaling access.

In [59] the use of sensor accelerometers is advised for the generation of high quality 128-bit random numbers. Such approach is a True Random Number Generator (TRNG) which offers perfect randomness for key generation.

2.5 Hybrid

From the previous sections, it can be concluded that the use of PKC is too resource expensive to perform cryptographic operations on every message computation. Still, it seems to be a reliable idea to use it for key exchange with DH and follow the communication using a symmetric key[32]. [32] proposes a modification of Huang et al.[21] which solves the handshake in five instead of six rounds but has weaknesses in the authentication of the sensors. [35] also proposes a hybrid scheme that uses ECC, IBC and *tate pairing* for symmetric key distribution with a hardware accelerator and achieving impressive results.

Yoo et al.[62] refer to the use of ECDH for key establishment using the same key pool on all the nodes. Each node selects a key from the pool and a random nonce, the combination of the data permits to derive a session key.

In this chapter several tendencies to solve the key distribution problem have been presented, in the referenced sources the reader may find energy consumption values for some of the implementations. Despite the optimal use PKC many assume that it is only feasible using hardware accelerators. If this may be true in many situations, the microprocessor manufacturers are producing more capable chipsets with lower power consumption opening the door to possible software implementation of hybrid PKC-Symmetric Key Cryptography (SKC) schemes.

Chapter 3

Environment

Within this chapter, the boundary and the tools used in this work are described. First, an overview on the standards is given. Subsequently, the Contiki operative system network stack structure is described, for which the key derivation and distribution solution will be developed. Finally, the available testing hardware is presented.

3.1 IEEE 802.15.4g

IEEE 802.15.4 is a standard that specifies the protocol and interconnection of devices via radio communication in a Personal Area Network (PAN)[2]. It makes use Carrier Sense Multiple Access (CSMA) and supports star and P2P topologies. The standard defines the physical layer and the Media Access Control (MAC) layer for device intercommunication.

The original standard of 2003 defines three operation frequency bands: an European band at 868MHz at 20Kbps and BPSK modulation, an American band at 915MHz at 40Kbps with the same modulation and a worldwide band at 2450MHz at 250Kbps and O-QPSK modulation. However, in the Amendment 3 new bands where defined, allowing 50Kbps and O-QPSK at 868MHz band[3] and specifying a Common Signaling Mode using F-2FSK modulation at 50kbps in all the bands.

The standard defines 64-bit addresses with the possibility of transmitting 16-bit addresses and using address expansion (as explained in Section 3.1.1) for obtaining the equivalent 64 bit address. This is a big advantage as the maximum frame size described is 127 bytes.

The MAC frame is reproduced in Figure 3.1 where the size variance of some of the fields can be observed, this is due to the specification in the Frame Control field (Figure 3.2). For instance if the security flag is not set, the security field will not be included, and in case it is, it will depend on the configuration. The same applies for the PAN-ID. When a message is sent inside the same PAN, the Intra PAN flag is set and the PAN fields of the MAC frame are elided. The addressing fields vary depending on the length of the transmitted address (0, 16 or 64 bits) which is indicated in the Dst Mode and Src Mode in the Frame Control field as well.

0 1 2	3	4	5	6	7 8 9	10 11	12 13	14 15
Type	Sec	Pend	ACK	Intra PAN	Reserved	Dst Mode	Reserved	Src Mode

Figure 3.2: *IEEE802.15.4 Frame Control Field*

2	1	0,2	0,2,8	0,2	0,2,8	0-14	variable	variable	2
Frame Control	Sequence number	Destination PAN-ID	Destination Address	Source PAN-ID	Source Address	Auxiliary Security Header	Information Elements	Payload	FCS

Figure 3.1: *IEEE802.15.4 MAC frame with field sizes in bytes*

IEEE802.15.4 specifies as well security suite options, being AES-CCM-128 the recommended one. This provides access control, data encryption, frame integrity and sequencial freshness. See Section 3.3 for more detailed information of AES-CCM-128.

Given the nature of WSN, the minimum amount of data is intended to be sent. When the nodes belong to the same PAN-ID the standard only produces 9 bytes of overhead in clear with an additional 4 bytes MIC after AES-CCM. 13 bytes overhead in total, allowing theoretically 114 bytes for payload.

3.1.1 Address Expansion

IEEE802.15.4 defines 64-bit addresses but it is also possible to use short 16-bit addresses, which have to be expanded to a "pseudo 48-bit address" and finally to a complete 64-bit address[30, 58]. This happens as shown in Figure 3.3.

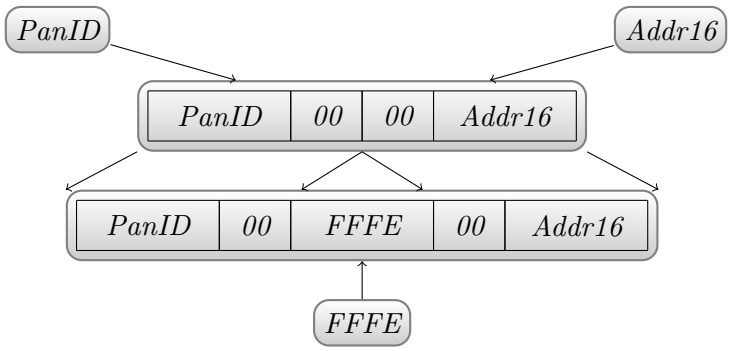


Figure 3.3: *EUI-64 address expansion from a 16-bit address*

This 64-bit address can be expanded again to obtain a fully IPv6 address by appending the prefix FE80::/64 and named LLA. In the rest of this document, the 64-bit address is referred as LLA since the difference with the real LLA is a fixed padding.

3.2 6loWPAN

6loWPAN is an acronym for IPv6 over low-power Wireless Personal Area Networks and appears from the idea of enabling IEEE802.15.4 networks to use IPv6. Since the IEEE802.15.4 standard generates small packets it makes it not suitable for the usage IPv6 on them, therefore some adaption layer is necessary.

The frame size defined in the 2003 standard[2] is 127 bytes in the physical layer, with a maximum header size of 25 bytes, leaving 102 bytes to the MAC layer. Considering an additional encapsulation overhead of a maximum 21 bytes due to AES-CCM-128 results in 81 bytes of maximum payload. At the same time the minimum packet size of the IPv6 standard is 1280 bytes, inclusive a header of 40 bytes. Additionally 8 bytes for User Datagram Protocol (UDP) headers or 20 bytes for TCP headers should be considered. This situation leaves between 20 and 33 bytes for upper layers data[30].

Given this problem, the 6loWPAN adaption layer defines a IPv6 compressed frame structure such as in Figure 3.4. Each header field of the frame contains a header type followed by zero or more header fields such as: addresses, hop-by-hop options, routing, fragmentation and payload.

Figure 3.4: *6loWPAN frame*

The dispatch header starts with a zero bit and a one as the second bit, then a 6-bit header selector and N bits for type specific header as in Figure 3.5. This allows to select which kind of header will follow. Examples are `000001` for full IPv6 header, and `000010` for loWPAN HC1 header.

In case the frames need to be fragmented –which is supported by IPv6– the fragmented header is appended. This header follows the structure represented in Figure 3.6. The red field *offset* is only present when the frame is not the first fragment while the 1 in the identifier field turns 0 for the first fragmented packet, being 1 for the rest. This situation is not attractive for WSN communication as produces a higher header overhead and leaving less room for the payload of the message.

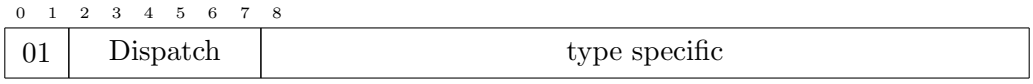


Figure 3.5: *6loWPAN dispatch header*

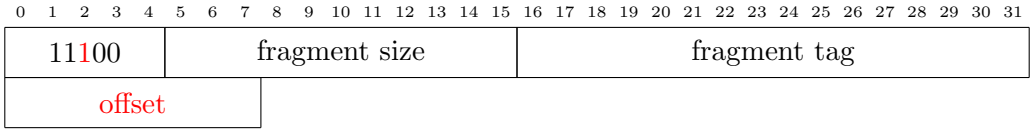


Figure 3.6: *6loWPAN fragmentation header*

3.2.1 Header Compression

The header compression is one of the most interesting features of the 6loWPAN standard. It makes it possible to send very few overhead for TCP/UDP over IPv6 communications. The first standard, RFC4944, defines two levels of compression: *HC1* and *HC2*. However, these are considered insufficient for most practical uses of IPv6 in 6loWPAN[22]. RFC6282 defines two more compression levels *IPHC* for effective compression of Unique Local, Global and Multicast IPv6 addresses; and *NHC* which defines an encoding for arbitrary next headers.

HC1

The main idea behind this compression is that the IPv6 packets will always take the same values for some of the fields:

- IP version is always 6.
- The addresses can be resolved from the MAC addresses.
- The frame length can be inferred from the IEEE802.15.4 frame or the size field in the fragment header (Figure 3.6).
- Traffic Class and Flow Label are zero.
- The next header is UDP/TCP or ICMP.

The HC1 byte is represented in Figure 3.7. PI fields identify how the addresses are transported (10 means compressed and in-line), TC/FL is the compression of the Traffic Class and Flow Label, the Proto field points out which is the protocol to use (TCP, UDP, ICMP or full specified in-line) and HC2 points out if a HC2 header is provided after HC1.

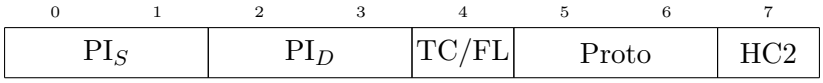


Figure 3.7: *6loWPAN HC1 byte*

HC2

When HC1 indicates UDP and the HC2 flag is set, the HC2 header must follow. The HC2 reduces the overhead of UDP from 8 bytes to 4 bytes. In Figure 3.8 a HC2 header is reproduced. The C fields indicate whether compression is enabled or not for the Source Port (SP), the Destination Port (DP) and the Length (L). The latter means that the length of the payload is derived from the payload length of the IPv6 header, rather than being again specified. The compression for the ports allows a port to be sent as a short value of 4 bits. This value is added to 61616 (0xF0B0) to obtain the final port number.

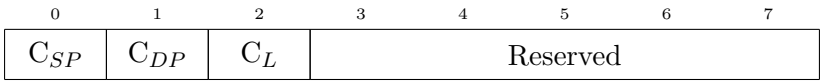


Figure 3.8: *6loWPAN HC2 byte*

An example of a whole UDP over 6loWPAN frame using HC1 and HC2 compression is reproduced in Figure 3.9. First, comes the dispatch header indicating HC1 loWPAN compression, the second byte is the HC1 header pointing out UDP as a protocol and to which follows a HC2 compression header. This indicates compression for ports and length. Finally, further IPv6 and UDP headers (minimum additional 6 bytes) are appended.

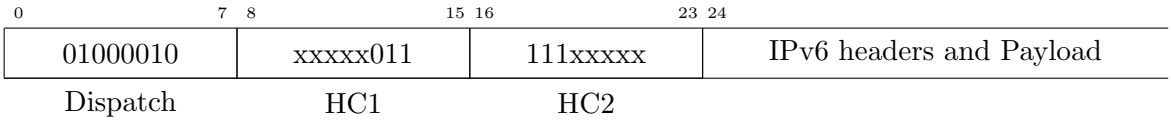


Figure 3.9: *6loWPAN compressed UDP frame*

Field	Bits	Description
TF	2	Traffic Class and Flow Label
NH	1	Next Header compressed using NHC or not
MLIM	2	Hop Limit compression
CID	1	Context Identifier Extension
SAC	1	Source Address Compression
SAM	2	Source Address Mode
M	1	Multicast Compression
DAC	1	Destination Address Compression
DAM	1	Destination Address Mode

Table 3.1: *6loWPAN IPHC word fields description*

IPHC and NHC

This new extension proposed in [22] compresses the IPv6 header even more and makes the 6loWPAN frame more generic for different succeeding headers using a NHC byte. This frame defines which is the next header to follow and indicates if it is compressed or not.

Figure 3.10: *6loWPAN IPHC header structure*

The fields of the IPHC byte are as described in Table 3.1. In Figure 3.11, the NHC frame is depicted and in Figure 3.12 the compressed header for UDP is presented. C stands for checksum, when it is 1, the checksum is elided and is recovered by recomputing it from the 6loWPAN termination point. P stands for the compression of the ports, they can be fully carried in-line, 8 bits elided and 12 bits elided. This header compression offers more flexibility than HC1 and HC2.

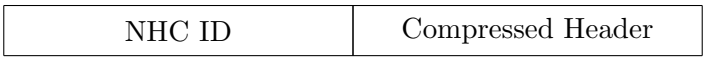


Figure 3.11: *6loWPAN NHC structure*

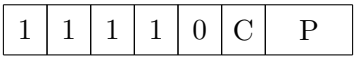


Figure 3.12: *6loWPAN UDP compressed header structure*

Figure 3.13: *6loWPAN Full UDP packet using IPHC and NHC*

The standard defines 2-3 bytes for IPHC header and 1 byte for each NHC. The different fields in IPHC allow a specification of the size of the source and destination addresses as 128-bit, 64-bit or 16-bit addresses or fully elided (and computed from the encapsulating header). This structure reduces to 7 bytes the full UDP/IPv6 packet headers when addresses are elided. This is shown in Figure 3.13 where the red fields represent optional fields such as additional IPv6 headers, addresses and fragmentation in case of a large frame.

It is easy to deduce then, that the overhead for a UDP/6loWPAN over IEEE802.15.4 packet produces only between 14 and 28 bytes for the header. In Figure 3.14 it can be seen that the use of UDP on broadcast addresses using 6lowPAN and 64-bit addresses for IEEE802.15.4 maximizes the size of the payload.

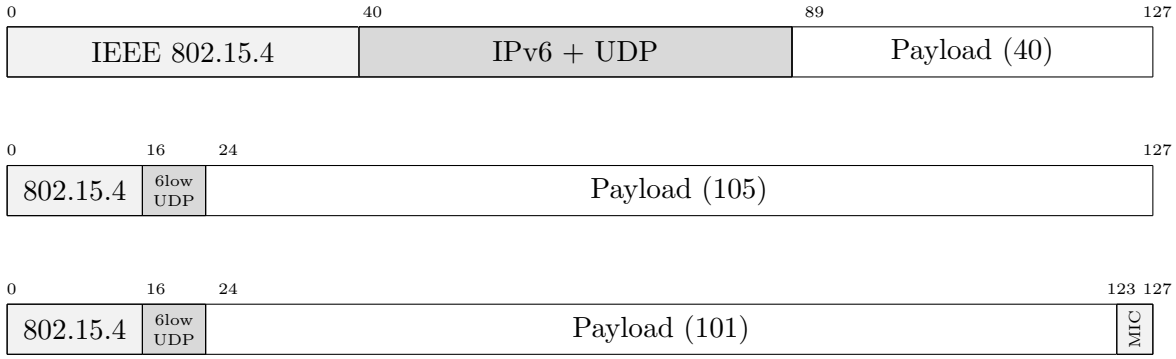


Figure 3.14: *Comparison of frames for IEEE802.15.4, 6loWPAN and AES-CCM MIC. On top, IEEE802.15.4 with full IPv6 and UDP headers without compression. The second frame uses compression for broadcast addresses and 6loWPAN using IPHC/NHC. The third frame adds an AES-CCM-128 4 bytes MIC.*

3.3 AES-CCM 128 bits

CCM is an operational mode for block ciphers such as AES and stands for Counter (CTR) with CBC-MAC. It provides an authenticated encryption: integrity, authentication and confidentiality and is described for use with AES in RFC3610[61]. The standard defines four different inputs: the encryption key, a nonce, the message contents and authentication data.

For integration within IEEE802.15.4, its header is used as a 13-bit authentication data while the first four bytes of this data are meant to be the nonce of choice. This is a good practice since a multicast IEEE802.15.4 frame has already 13 bytes of header, variable depending on the source and destination addresses, giving enough information to ensure its authenticity in the communication. The sequence number of IEEE802.15.4 offers at the same time a simple solution for the nonce without more overhead. The key is only known by the two communicating ends, therefore it is safe to use this data for authenticating, validating integrity and confidentiality. The chosen size of the MIC is 4 bytes which offers a good trade-off between overhead and security as detailed in Section 6.1. The resultant frame is represented in the bottom frame in Figure 3.14.

The procedure works as follows: First the CBC-MAC of the data is computed to obtain a MIC and then the CTR of the payload. Then, the cleartext header is sent together with the encrypted payload and the MIC to the other end.

3.3.1 CBC-MAC

Firstly, a CBC-MAC code is computed over a flags byte, the nonce, the authentication data and the payload. This allows a MIC which provides data integrity and authentication.

Figure 3.15: *AES-CCM flags byte*

The flags byte (Figure 3.15) is formed by specifying the sizes of the MIC and the length field and indicating if authentication data if provided. The nonce is appended to this byte together with the authentication data and the payload. The result is a 128-bit block of encrypted data which is the MIC of the packet. The block chaining for a CBC-MAC operation is depicted in Figure 3.16.

Once the MIC is obtained, it is prepended the same flag frame, with A *len* and A set to 0. This is encrypted and truncated to obtain a new encrypted MIC of 32 bits which the receiver can validate providing integrity and authenticity. A 32-bit MIC is considered secure enough as discussed in Section 6.1.

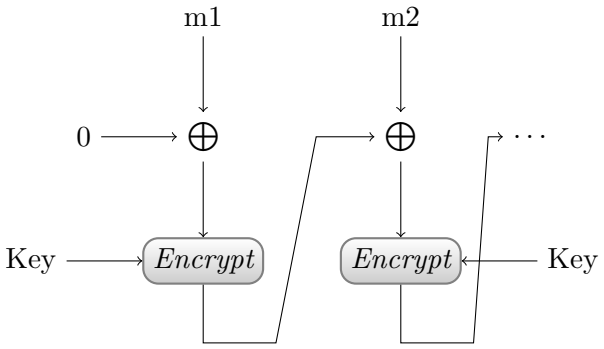


Figure 3.16: *CBC-MAC encryption*

3.3.2 CTR

The second step of the procedure is to encrypt the payload using a counter fashion. This is sketched in Figure 3.17. The concatenation of the nonce and a counter is encrypted and XORed with the plaintext, producing the ciphertext.

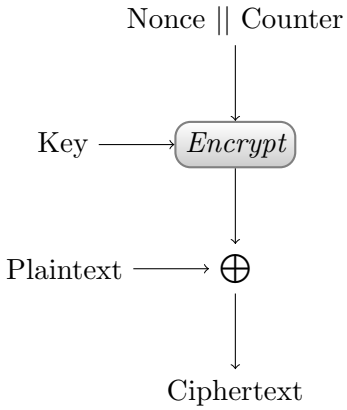


Figure 3.17: *Counter Encryption Chain*

3.4 Contiki

Contiki is *the open source operating system for the Internet of Things*[1] developed by Adam Dunkels at the Swedish Institute of Computer Science (SICS). It is fully written in C language and is able to run on different hardware platforms. It uses full low-power Internet

communication while being compliant of the most recent wireless standards such as Routing Protocol for Low-Power and Lossy Networks (RPL), 6loWPAN, IPv4, IPv6 and IEEE802.15.4 among others.

At the same time its relatively easy to write programs for Contiki thanks to the Protothreads library and the ever growing community behind this project. Since July 2012, Contiki creator and main developer Adam Dunkels, founded Thingsquare which offers private support and integration of Contiki for private users and companies.

3.4.1 Memory Footprint

Even though the Contiki website claims a small memory footprint for an RPL sleepy router with less than 30KB of ROM and 10KB of RAM, these numbers are not suitable for some applications (see Section 3.5) as the most memory constrained network has just 32KB of ROM and 4KB of RAM. However, this can be solved as Contiki is highly configurable and offers by default the possibility to trim down the memory consumption by setting some compiling time macro definitions. It has been reduced to use just 18KB of ROM and less than 3KB of RAM performing good in a 3 nodes network. Some of the changes have been committed to the Contiki project for public benefit.

3.4.2 Network Stack

Contiki is very modular, its network stack (Figure 3.18) is a prove of it, which makes everything transparent for the programmer.

When an application requires to send data over IP, this data and destination information is passed along to the μIP stack, a fully IPv4 and IPv6 stack. There, the full TCP, UDP or ICMP packet is built up and embedded in an IPv6 packet. Subsequently, the IPv6 packet is passed to the 6loWPAN adaption layer which compresses the headers according to the 6loWPAN standard (see Section 3.2). When the packet is ready and the MAC driver –CSMA in Tado case– considers that the medium is ready, an IEEE802.15.4 packet is built and the Radio Duty Cycle (RDC) controller takes care of it delivering it to the radio interface.

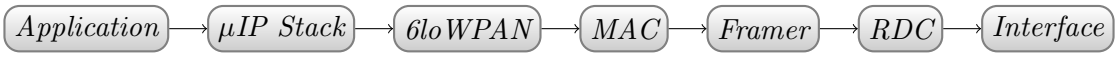


Figure 3.18: *Contiki network stack (sending chain)*

Figure 3.19: *Contiki network stack (receiving chain)*

The receiving chain differs in the sense that the RDC is in charge of turning the radio on and off. Therefore, the RDC sets the radio in listening mode and when a packet arrives it is first driven to the framer, in order to expand the information, the chain goes in reverse way: from 6loWPAN inflation methods, to μIP unpacking and IP checking, finally delivering it to the application.

3.4.3 Implementation

For the implementation of the proposed solution, two modules are developed according to Contiki's structure. One is the Key Manager Application which is in charge of the key

establishment routines, key renewal and generation procedures. The other module is a lower module, between the RDC and the Framer. This one is in charge of packet encryption and decryption depending on the link key. This is represented in Figure 3.20 and 3.21.

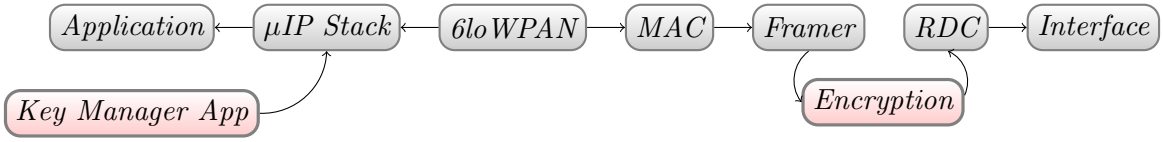


Figure 3.20: *Contiki network stack with hooks (sending chain)*

Figure 3.21: *Contiki network stack with hooks (receiving chain)*

3.5 MSP430, CC430, Cortex M3 and CC1101

The design of the protocol has been focused on the particularities of the hardware specified in Table 3.2. Nevertheless, it is easily portable to other platforms.

Tado's network is composed of three different kind of nodes with different resources (see Table 3.2). All the devices have compatibility with 32-bit storage, either native support as the Cortex M3 or emulation support for the other platforms.

From the table, it can be seen that the most constrained node for both computational power, memory and available energy is the CC430 platform, which is designed to run for two years without replacement of the batteries. Therefore, an additional effort is required in order to minimize the power consumption of such platform. This is achieved by using a reduced version of Contiki, without RPL and just capability for link-local broadcasts.

	GW	**CH**	**Sensor**
Processor	LM3S9997	MSP430F5335	CC430F5137
Operation frequency	50 MHz	20 MHz	4 MHz
Address map	32 bits	16 bits	16 bits
Flash memory	128 KB[1]	128[1] KB	32 KB
RAM	64 KB	16 KB	4 KB
Power	DC 5 V	HVAC supply	2 AAA batteries and a Solar Cell

Table 3.2: *Hardware description*

3.5.1 Cortex M3

The Cortex M3 is a well suited processor with 256 KB of Flash and 64 KB of RAM that can run up to at 80MHz. However, the frequency is limited to 50MHz in order to decrease

[1] Albeit the real flash size is 256 KB, it is divided in two different partitions and therefore the final size of a partition is just 128 KB.

the power consumption as it is not necessary to operate at such a high frequency. Despite of lowering the frequency this platform has a higher frequency of the rest of the nodes to ensure the correct communication with the Internet.

Additionally, it has an internal ROM with the StellarisWare which contains AES SBOX tables, Ethernet drivers, CRC-CCITT peripheral and the most common ADC interfaces and peripherals.

In Tado's platform, this processor is used as a Gateway (Figure 3.22), connected via Ethernet wire to a home router which allows to bridge the IEEE802.15.4 network with the Internet.

The gateway is connected to the electricity plug continuously allowing better performance and keeping a stable input power supply. It uses a CC1101 RF chipset in order to communicate with the other devices of the network.

Figure 3.22: *Tado gateway (Cortex M3) mote*

Figure 3.23: *Tado cluster (MSP430) head mote*

3.5.2 MSP430

The MSP430 family is an ultra low-power microcontroller family, allowing wake-ups from low power modes to active modes in less than 3 μs and an operation frequency up to 20MHz. It operates at its maximum frequency as the power supply is guaranteed by the house power supply. On the other hand, the low-power consumption must be guaranteed for the goal of the application.

This platform is not as fully featured as the Cortex M3 but offers a really low power consumption and it is ideal for operating using the Heating, Ventilation and Air Conditioning (HVAC) power supply without interfering with the control operation (Figure 3.23).

The platform is connected to a CC1101 RF chipset for RF communications and uses a saving RDC algorithm with an on-time of about 6.25%.

3.5.3 CC430

The CC430 chipset is ultra low-power consumption microprocessor, while embedding a CC1101-like RF chipset. It offers lower power consumptions than the MSP430, at the same time that allows an operation frequency of 20MHz. This frequency is not required and therefore lowered to 4MHz to ensure even lower power consumption, which is lineally related to the operation frequency.

Additionally, it has an AES encryption/decryption peripheral embedded in spite of its flash memory size which is just 32 KB. The flash and RAM sizes make it extremely constrained for many applications, but its very low power consumption turns it into a perfect microprocessor for battery operated devices such as the Tado sensor (Figure 3.24), which reports temperature periodically.

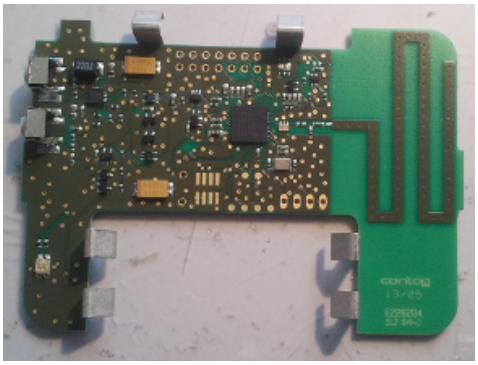


Figure 3.24: *Tado sensor (CC430) mote*

3.5.4 CC1101

The CC1101 is a sub-1GHz transceiver designed for very low-power applications, which makes it suitable for IEEE802.15.4 operation on the European band (868MHz) supporting different modulations and data rates up to 600kbps[1]. The datasheet claims 200nA of current consumption in sleep mode, 11mA in receiving mode and 22mA in transmitting mode, which makes it an ideal candidate for any device in a WSN.

As it can be observed in Section 6.4, the CC1101 chipset offers very low variance of the current consumption for different supply voltages, making it extremely attractive for this application.

[1]Note that the default Common Signaling Mode at 50kbps is used for the implementation

Chapter 4

Key Generation

In this chapter, an overview on all the necessary background and components for key generation is given. The recommended procedure for key derivation is ECDH which is described in Section 4.3.2. Other procedures regarding challenge solving and hashing algorithms are also reviewed in this chapter.

4.1 Random Number Generation

Two approaches are generally accepted when speaking of random number generation: Pseudo Random Number Generator (PRNG) and TRNG. The latter provides randomness through physical phenomenons that are difficult to predict such as number of radioactive particles in the ambiance or the noise received with the radio signal. On the other hand, PRNG use mathematical algorithms to compute "random" values from a given seed, or initial value.

One of the objectives of a WSN is that devices on the field are as much energy efficient as possible and the sensing option may have too high energy consumption for an extra use rather than the normal purpose of the mote. The use of TRNG to generate a seed to feed a PRNG (see Section 4.2) is recommended. If the quality of the PRNG is proved, then *good* seed suffices for providing the system satisfactory random numbers.

Several algorithms have been considered, Relic-Toolkit random number generator, Mersenne Twistter and glibc random implementation among them. However, given the memory constraints in the CC430 (see Section 3.5) the default glibc PRNG was used for the implementation. It is trusted for being an open source implementation and tested by many. It has a really small footprint of about 38bytes for *rand*() and 10 bytes for *srand*() functions on the CC430 binary. As already mentioned, the strength of this implementation bases on the randomness of the seed, and therefore the PRNG must not be initialized twice with the same seed. For the seed generation, the different mote capabilities have been used in order to provide as much randomness as possible to the seed, which is described in the following sections.

4.2 Entropy Collection

It is trusted in the strength of glibc PRNG given a good seed. The issue arising is in which way this seed should be generated to ensure sufficient randomness. The advantage of a sensing node in front of any other networking node is that physical information of the environment

can be collected in order to generate a random number. For instance, the temperature gradient between the boot time and some seconds later will depend in atmospheric and environmental characteristics as well as manufacturing procedure, leading to differences among similar devices. Another unpredictable information is the number of bytes sent or received by a radio interface as it depends on the neighboring communications and interferences. The time the device was running and the remaining battery life are other examples.

The information is salted using the serial number of the devices, hence two devices of the same kind will give different seeds under the same conditions. The use of the serial number rather than the LLA is due to the fact that the access to the serial number implies physical access to the device as it is only printed in the boxing.

4.2.1 Entropy Collection in Tado motes

The sensors forming the Tado network are not identical, thus the same algorithm cannot be used to collect entropy in all of them. In the following subsections the algorithm used in each device is described. For that means, the auxiliary functions $D(x, y)$ defined in Equation (4.1) and $H(x)$ are used. D converts two bytes into a word variable and H is a general purpose hash function (see Section 4.5).

$$D : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N} \tag{4.1}$$

$$D(x, y) = x \cdot 256 + y = x \mid\mid y \tag{4.2}$$

$$H : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N} \tag{4.3}$$

Gateway

The GW mote has no sensors. However, it has an Ethernet port and a RF chipset which can provide statistics of the number of received or sent packets (RP_E, RP_{RF}). Additionally, the number of active processes in Contiki (N_{proc}) is also interesting to use as it depends on the firmware and the device activity. The reason of the last reset (R) (watchdog, remote update, strange device state or AC failure) is not always the same, thus it is compelling to use as well as the device serial number (SN) and the running time (T_{run}). Equation (4.4) defines the value of the random seed for this platform.

$$H(D(RP_E, RP_{RF}) \oplus T_{run} \oplus (SN + N_{proc} + R)) \tag{4.4}$$

Cluster Head

The cluster head in Tado's network is connected to the voltage supply offered by the HVAC system which differs depending on the model, brand and house AC network characteristics. The seed is computed as in Equation (4.5) where ΔT_{low} stands for the lower byte of temperature reading delta (the centigrades) between the device boot temperature and the temperature in the requested moment; and V_{HVAC} is the reported input voltage from the HVAC. The rest of variables are as for the Gateway case.

$$H(D(RP_{RF}, \Delta T_{low}) \oplus T_{run} \oplus (SN + N_{proc} + R) \oplus V_{HVAC}) \tag{4.5}$$

Temperature Sensor Node

The temperature sensor has a temperature sensing chip, an RF chipset and peripheral buttons. The Equation for the temperature sensor is as in 4.6.

$$H(D(RP_{RF}, \Delta T_{low}) \oplus T_{run} \oplus (SN + R + B_{cycles})) \tag{4.6}$$

In this case, B_{cycles} stands for the number of processor cycles during which the pairing button was pressed.

4.3 Elliptic Curve Cryptography

ECC is an approach to PKC based on elliptic curves over finite fields introduced in [27]. An elliptic curve is defined by the Equation (4.7) and a *point at infinity*. An example of an elliptic curve over $\mathbb{R}$ is reproduced in Figure 4.1. Some research outcomes claim that 163-bit ECC provides a similar security to an RSA with a 1024-bit key[48].

$$y^2 = x^3 + ax + b \tag{4.7}$$

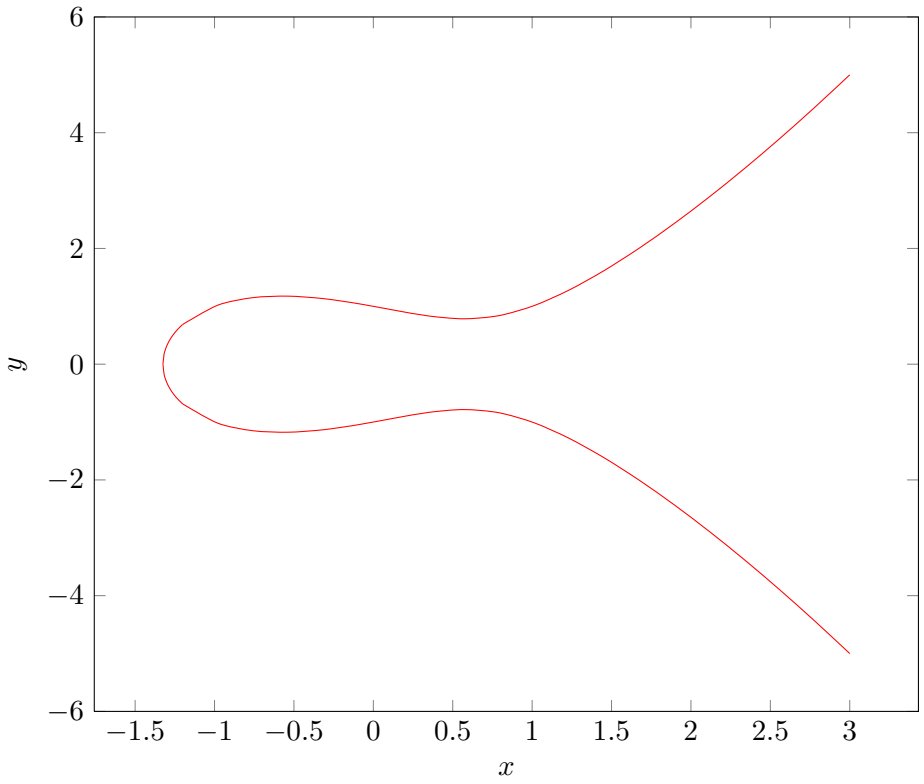


Figure 4.1: *Elliptic curve equation* $y^2 = x^3 - x + 1$

A curve is defined by several parameters (q, a, b, G, r ,h), q is the field, a and b are the coefficients of the curve, G is the generator or base point, r is the order of the generator, and h is the cofactor. The security of ECC bases on the ability to compute point multiplication, and the inability to compute the multiplicand given the original product of points (see Section 4.3.1 and 4.3.2): this is exactly the ECDLP.

4.3.1 Elliptic Curve Discrete Logarithm Problem

As already mentioned, the Elliptic Curves Cryptography bases its strength on the difficulty to solve the ECDLP which derives from the traditional DLP stated in Definition 1. ECDLP is stated in Definition 2.

Definition 1. *Let G be a multiplicative group and $g \in G$, let $\langle g \rangle$ be a cyclic subgroup generated by g. The Discrete Logarithm Problem for the group G can be stated as:*
Given $g \in G$ and $a \in \langle g \rangle$, find an integer x such as $g^x = a$.

It is very difficult to find an integer x that fulfils this equation in a cyclic group since many are the possibilities. This is the base for the DH Key Exchange.

Definition 2. *Given an elliptic curve E over a finite field $\mathbb{F}$. Let P and Q be two points on this curve, find $d \in \mathbb{Z}$ which fulfills the relation $Q = d \cdot P$.*

For the particular case of elliptic curves, P is a commonly known point, d is a secret factor or *Private Key* which is only known by the owner, and Q is a publicly known point or *Public Key*. d and Q are referred as *key pair* in the rest of the document.

4.3.2 Elliptic Curve Diffie-Hellman

DH is a key exchange method which allows two entities to establish a common secret over an insecure channel and it is named ECDH when operating over elliptic curves. Two entities using the same P point (as in Definition 2), which depends on the curve and is usually the curve generator (G), compute a common key K such as in (4.10).

$$d_A \in \mathbb{Z} \qquad d_B \in \mathbb{Z} \tag{4.8}$$

$$Q_A = d_A \cdot G \qquad Q_B = d_B \cdot G \tag{4.9}$$

$$K = d_A \cdot Q_B = d_A d_B G = d_B \cdot Q_A = d_B d_A G \tag{4.10}$$

This means that two entities operating on a common curve can exchange the points Q_A and Q_B in order to obtain a common point K which will be different for each pair of points Q_i. Hence, only the entities that know this points can obtain the same common point on the curve.

4.3.3 Koblitz Curves

Koblitz curves, also known as anomalous binary curves, refers to a family of curves over the field $\mathbb{F}_2$[28] with the advantage that the coefficients are optimized for efficient arithmetic operations. Koblitz curves follow the Equation (4.11).

$$y^2 + xy = x^3 + ax^2 + 1 \tag{4.11}$$

Recent research proves that even the computation over Koblitz curves is optimized thanks to the Frobenius endomorphism, which is also the cause of a security issue on these curves[17]. For the Curve of our concern theoretically 2^{81} operations are required to break an elliptic curve over the field $\mathbb{F}_2^{163}$ (Equation (4.12)). However, only 2^{77} are actually required (Equation

(4.13)). Despite of this fact, it is also mentioned in [17] that *there is no evidence that elliptic curves with small class are insecure for cryptography.* Therefore, for memory and computation constraints we will use a NIST Koblitz 163-bit curve (see Section 4.3.4 *NIST K-163 curve*) with a class of 1.

$$\sqrt{\frac{q\pi}{2 \cdot h}} \approx 2^{81} \tag{4.12}$$

$$\sqrt{\frac{q\pi}{4 \cdot 163 \cdot h}} \approx 2^{77} \tag{4.13}$$

4.3.4 NIST K-163 curve

National Institute of Standards and Technology (NIST) issued a document in 1999[44] with a collection of recommendations for the choice of curves and cryptographic key lengths for governmental use. Due to some hardware limitations, the Koblitz curve K-163 defined over a 163 degree binary field ($q = \mathbb{F}_2$) is used. It is defined by the following parameters:

$$h = 2 \tag{4.14}$$
$$p(t) = t^{163} + t^7 + t^6 + t^3 + 1 \tag{4.15}$$
$$a = 1 \tag{4.16}$$
$$r = 5846006549323611672814741753598448348329118574063 \tag{4.17}$$
$$G_x = 2fe13c0537bbc11acaa07d793de4e6d5e5c94eee8 \tag{4.18}$$
$$G_y = 289070fb05d38ff58321f2e800536d538ccdaa3d9 \tag{4.19}$$

Where a is the coefficient; r the base point order; $G_{x,y}$ are the coordinates x and y of the generator and h is the cofactor.

4.3.5 Relic Toolkit

Relic Toolkit[4] is a mathematical library with emphasis on efficiency and flexibility. It is chosen instead of libTomCrypt taking into account the analysis in [49] and the ease of integration into a Contiki environment. Both are Open Source projects and have been reviewed and used by many, hence they are considered trustworthy.

Relic offers a complete cryptographic library with already implemented point operations for elliptic curves over binary fields, and optimizations for Koblitz curves, which are the ones of our interest. It provides a high level of customization and different methods for curve point generation and multiplication. Despite of its modularity, modifications were carried out to embed the library in the most constrained platform (CC430) by limiting its features to just one curve and some modifications in the random number generator and in the SHA1 library.[1]

Relic permits to choose between different operational algorithms (e.g. several point multiplication approaches). Table 4.1 shows the memory and speed deltas for the chosen methods. These methods are chosen as they offer a fair trade-off between processing and memory overhead. With the basic configuration, the generation of the points took more

[1] According to it's license (LGPL) all modifications have been published in this Theses Author github account, on a fork of the relic repository in `https://github.com/lebrush/relic-toolkit`.

than five minutes of computation on the CC430. From that starting point, the different configurations were tested and discussed.

Module	ΔT	ΔS	**Description**
Point Multiplication Method	33 s	+ 2560 bytes	The basic multiplication has been replaced by the López-Dahab multiplication[33].
Binary Field Multiplication Method	- 11 s	- 210 bytes	Replaced the basic multiplication for the López-Dahab method.
Binary Field Inverse Method	- 2 s	+ 250 bytes	Selected the binary method for inverse calculation.
Binary Field Modular Reduction Method	- 14 s	- 294 bytes	Faster modular reduction method.
Binary Field Squaring	- 12 ms	- 198 bytes	Setting a table based squaring method reduces the computation time and the memory footprint.

Table 4.1: *Relic performance and size with several configurations (CC430)*

After the modifications and optimizations, the library memory footprints on the different platforms (see Section 3.5) are described in Table 4.2. The values in the table represent the memory footprint of relic with the same configuration before the code optimizations and after the code optimizations, the goal of the code modification led to huge memory footprint reductions.

Microprocessor	**With Point Generation**	**Without**	**SHA1**
Before optimization			
Cortex MX	8224	6774	784
MSP430	12004	9840	2734
CC430	10110	8348	1904
After optimization			
Cortex MX	4986	3412	720
MSP430	7916	5682	2554
CC430	6338	4446	1826

Table 4.2: *Relic memory overhead (in bytes) for the different processors*

4.4 Key Derivation Function

A Key Derivation Function (KDF) is a function which extracts one or more keys of a certain length from a given master secret. The function of choice is the KDF2 defined in ISO-18033[16]. This function is already implemented in Relic Toolkit. The procedure to obtain a key consists in hashing the input secret together with a counter, and iterate it until the desired key length has been achieved, increasing the counter in every loop and appending the result to form the resulting key. The algorithm is as follows:

1. $d = \lceil \frac{key\ length}{hash\ length} \rceil$

2. For $Counter$ =0 to $d-1$

   C = IntToString($Counter$)

   $T = T$ || Hash(secret || C)

3. The Key is the first n bytes of T as defined by *key length*

In our case, the master secret is the common point on the elliptic curve as in Equation (4.10), and the derived key is the link key.

4.5 Hashing Algorithms

A hash function is an algorithm that transforms a variable amount of data into a value of a fixed length. They have huge importance in cryptography as they are unidirectional functions. Thus, given the hash value of some data, the original data can not be retrieved back. It is highly important for instance for key derivation (Section 4.4) as small changes in the input can lead to completely different output hashes.

In example the SHA1 hash for "kth" is `6eeab79f9f 101d2c8909 ca87f7ab67 a8da21901c` while the same function over "ktH" gives as a result `46ab3a049f 2af3b7311b 49a610390c f890a7c25f`.

Three different algorithms have been considered as candidate hash functions for the embedded application. *SHA1*, *Quark*[5] and *Photon*[19]. The comparison results are shown in Table 4.3.

Quark is presented as a lightweight hash function, offered in four different configurations with different strengths from 64 to 122-bit security. Photon is also a lightweight hashing algorithm with focus on RFID security and inspired in the S-BOX structure of AES. Both Quark and Photon are considered lightweight hash functions: the source code compared to the code memory footprint of SHA1,is a proof of it. Despite of this, u-quark requires about 7KB of RAM memory for computing permutations, which makes it not a suitable candidate for many embedded applications.

Algorithm	**Digest size**	**Code size**	**Comments**
SHA1	20	1886	Standardized hash function
u-Quark	17	1271	Requires 7.2KB of RAM to perform the permutations. Therefore not suitable as the smallest sensor has about 4KB of total available RAM
Photon-128	16	1562	Smaller hash and smaller memory
Photon-160	20	1601	Good trade-off memory security

Table 4.3: *Comparison of hashing algorithms. Sizes are shown in bytes*

The provided level of security is similar between all of the analysed algorithms. And if a closer look into our purpose of the hash functions is taken, we can rapidly see that then are all suitable. The main use of the hash function is to be used in the KDF (see Section 4.4)

to generate an AES-128 key (16 bytes), to randomise the seed for the PRNG (2 bytes) (see Section 4.2) and to create message signatures and challenges (4 bytes). Therefore, a 16 bytes hash (or digest) is enough to fulfil this requirements, as any other size will be truncated. On the other hand, it is important that the algorithm strengths and weaknesses are known, as the most important security functions rely on it, and that it can be easily ported to different platforms. Hence, and in spite of the memory overhead, SHA1 is the chosen hash function.

4.6 Keys and Scopes

The proposed solution requires four different keys with four different scopes. Each of them serves one purpose[64]. The different keys are described below.

4.6.1 Secret Key

This 128-bit key is generated by the provisioning station and pre-loaded to the node during manufacturing together with the serial number and other unique information. Its main purpose is to provide authentication as a proof of identity and it must be kept secret. It allows encrypted communication with the server if necessary: i.e. provides end-to-end security. This key is shared between the node and the Trusted Authentication Server (TAS) and never renewed. Hence, if this key is corrupted or disclosed the node must have been tampered or harmed and the network must not authorize it anymore as a valid member. An Intrusion Detection System (IDS) should be present in the network to expose this situations. See Section 6.1 for detailed information about resilience.

In Tado, the communication with the server uses HTTP requests with a binary AES-CCM-128 encrypted payload. This key is used to encrypt these requests since both parties know the key.

4.6.2 Key Pair

The key pair, as mentioned in Section 4.3.1, consists of a integer number d and a point on an elliptic curve $Q = d \cdot G$. It is generated by the provisioning station and preloaded during manufacturing together with the secret key and the other unique information.

The reason it is not generated in the motes is because of the high power consumption due to long time microprocessor usage as described in Section 6.3 and the memory overhead due to it (about 2KB according to Table 4.2).

4.6.3 Link Key

The link key is the key to secure a communication link between two nodes. It is generated by the nodes using ECDH for key exchange. The two random challenges are appended to the calculated common secret and hashed using the KDF to obtain a link key as in (4.20). In case of link key renewal the key is different thanks to the randomness of the challenges.

Since every link holds a different key, a compromised node may only disclose the keys for its neighbouring links, not compromising the entire network.

$$K_{link} = KDF2(d_A d_B G \;||\; Challenge\ 1 \;||\; Challenge\ 2) \tag{4.20}$$

4.6.4 Group Key

The group key is used to encrypt multicast messages and grants a node access to the network and to automated pairing (see Section 5.7) with other nodes. A node is not granted the group key until it has been successfully authenticated.

The group key derivation is done by the CH. [46] debated the problematic of the group key generation. However, in the proposed scheme, as already mentioned, it is trusted in the strength of the PRNG and in the strength of the generated link keys. These can be trusted as they are generated using ECDH and two random challenges generated by two different nodes using a trusted PRNG.

When the group key needs to be renewed, the CH generates a random number and XORs its hash with all the link keys available in the CH. This value is passed to the KDF to obtain a new key from it (4.21). If the key is not different from the old one, a new number is generated and the procedure is repeated. The random number is XORed to the link keys to add an additional level of randomness since the link keys are influenced by random numbers generated by other entities. This influences the randomness of the new random group key by N external nodes and is therefore not easily predictable.

$$K_G = KDF2(H(\text{random}) \oplus (\bigoplus_{i}^{N} K_{\text{i-link}})) \tag{4.21}$$

Group Keys are only generated either when a node joins the network, leaves or has been compromised.

4.7 Challenges

A cryptography challenge is usually a number used only once (nonce), which allows an entity to authenticate another entity by means of its capacity to solve the mentioned challenge. The chosen challenges size is 32 bits. Although it is relatively small compared to the key size and target security, it offers a good trade-off between overhead and security achievements. Please refer to Chapter 6 for details about challenge size decisions and analysis.

For a node to authenticate another node and provide strong authentication, simple encryption and hashing chaining schema is used. Imagine Node A (the newcomer) and Node B want to authenticate each other. Node A will send a random 32-bit number to B and the other way round.

Node A receives this challenge, hashes it to obtain a 16 bytes hash, then encrypts the result using the recently generated link key, truncates it to 32-bit integer and hash it once more to obtain a 16 bytes string. Finally, it encrypt the hash one last time using its secret key and finally truncate the hash of it to 32-bit integer one last time.

When node B, which is already part of the network, receives a challenge, it hashes the challenge and encrypts it with the link key. Finally, it hashes the result once more and truncates it to a 32-bits value, which is sent to the TAS. The TAS hashes it once more, encrypts it with the node secret key and returns a 32-bit truncated hash of it to Node B.

This procedure is repeated for both challenges, so that both nodes (A and B) have the solution of the two challenges and can authenticate each other. In (4.25) the mathematical challenge solution is summarized. $H(x)$ is a hashing function which produces digests between 128 and 160 bits, $T(x)$ is a function that truncates the data to 32-bit chunks, and $E_k(x)$ is

the encryption function for the cleartext x using the key k.

When node A provides the result to node B, this one knows A is a genuine node willing to join the network. When node B provides the final solution to node A, the latter authenticates the network.

$$H : \mathbb{N} \longrightarrow \mathbb{N} \tag{4.22}$$

$$T : \mathbb{N} \longrightarrow \mathbb{N} \tag{4.23}$$

$$E_k : \mathbb{N} \longrightarrow \mathbb{N} \tag{4.24}$$

$$\text{Solution} = T(H(E_{\text{secret}}(H(T(H(E_{\text{link}}(H(\text{challenge}))))))))) \tag{4.25}$$

It can be seen that the challenge solving is done by two identical operations: $T(H(E(H(x))))$ which only differ in the key to use. These are called the challenge partial solutions or *partials*.

The fact of hashing the encryption results before truncating them is done in order to avoid any possible correlation between the cleartext challenge sent on the insecure channel and the value of the transmitted data between the CH and the TAS.

This challenge solving method using partial solution on the CH and the TAS allows to avoid MiM attacks as described in Section 6.1.

4.8 Signatures

A signature has the purpose to be the proof of the authenticity and integrity of a message. In the proposed protocol, the signatures are computed over the entire message setting the signature field to 0 and then replacing it by the value of the signature. First, the a hash of the message is calculated and then encrypted using the key shared with the other communication end. Finally, it is hashed and truncated again in a similar fashion as the challenge solving method. The proposed signatures method takes $6.8ms$ for 100 bytes in an MSP430, which is 130 times less than an ECC signature as mentioned in [47].

Signatures are used in pairing and in authentication. In the pairing process the signatures are computed to verify that the computed key by both entities is valid, therefore the link key is used. In authentication, the signature is used as an integrity verification of a certain authentication request message to avoid possible reply messages and computed using the secret key.

$$Signature = T(H(E_K(H(message)))) \tag{4.26}$$

Chapter 5

Key Distribution

In this chapter the way how nodes join and leaves the network and is revoked is described.

For this purpose a lightweight protocol over UDP is developed. The protocol frames follow the structure as in Figure 5.1. The header fields are as described in Table 5.1.

Within the following sections, the different procedures and the message exchange are described.

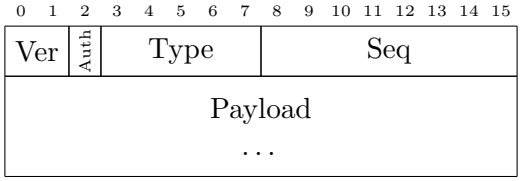


Figure 5.1: *General protocol message (see Table 5.1)*

Field	**Length (bits)**	**Value**
Version	2	Version of the protocol
Authentication	1	This flag is set to one whenever a node has not been authenticated by the network yet or vice versa
Type	5	The type of message being transmitted
Sequence Number	8	Incremental sequence number. Even though this protocol is meant to run over UDP which already includes a sequence number, it is interesting sometimes to ensure just application level security
Payload	-	Contents of the message

Table 5.1: *General protocol message description (see Figure 5.1)*

An overview of the protocol flow is represented in Figure 5.2 while a detailed version can be found in Figure 5.7.

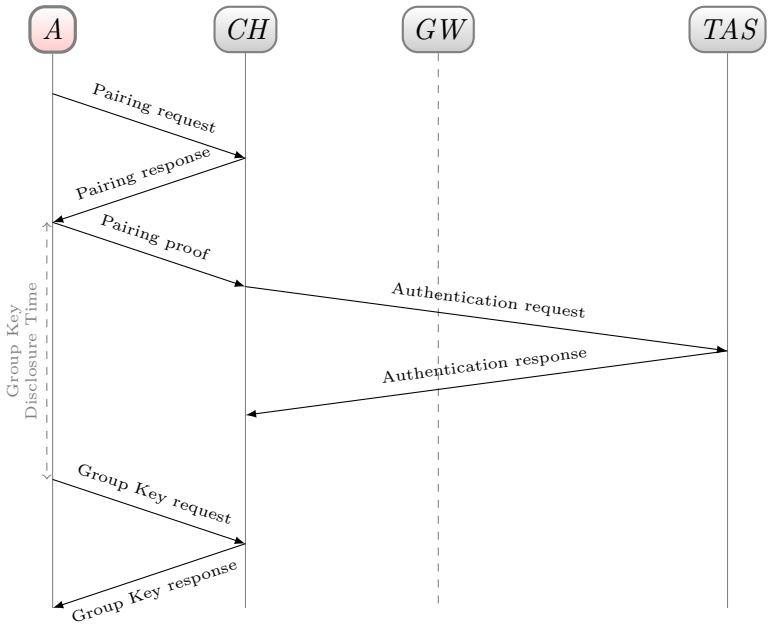


Figure 5.2: *Key exchange flow*

5.1 Provisioning

Some[55, 57] considered the provisioning phase as part of the key distribution phase, as it involves key generation process. For the proposed schema, a trusted KDS is required. The provisioning computer responsible for loading differentiation data into the nodes (Serial number, IP address, stock configuration,...) is a suitable candidate. The mission of the KDS is to generate and pre-load the private keying information into the nodes. This KDS generates a key pair (d_i and Q_i) and the secret key K_{S_i}. The former is pre-loaded into the nodes and immediately erased from the KDS memory while the latter is stored in the nodes and additionally shared with the TAS. The use of the KDS allows the motes not to compute the key pair, which speeds up the communication bootstrap for slow nodes.

Trust on the KDS is mandatory. The trust also applies to the fact that KDS distributes the keying material to to the nodes and to the TAS in a secure way.

5.2 Pairing

Pairing (or association) is the process of creating a secure channel between two previously unassociated devices over an insecure communication channel. In WSN the channel is the wireless medium and therefore vulnerable to Man-in-the-Middle attacks, eavesdropping and routing attacks among others.

This section describes a pairing protocol which allows a new node to associate with all the neighbours almost automatically requiring the minimum user interaction and very-low overhead making it suitable for any WSN. It uses ECDH for key exchange, SKC for securing the communication and a TAS to ensure strong authentication and authorization. The key management is meant to be built over UDP for compatibility, scalability and overhead. As it was described in Section 3.1 and 3.2, the characteristics of UDP multicast messages offer the lowest possible overhead over IPv6.

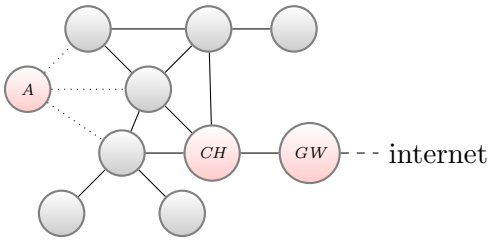


Figure 5.3: *Example of a unpaired node*

5.2.1 First Pairing: Joining the Network

This situation is given when a new node wants to join the network for first time. For the simplicity of the description, let a new node be *Node A*, which is willing to join a network such as illustrated in Figure 5.3.

All the nodes must pair with the CH as it is considered a node with more resources and it is the only entity with the right to renew the group key. To start a pairing process, the operator in charge of deployment must press the pairing button of Node A and the CH, which of course is already member of the network. This provides a *Proof of Proximity*[34], and makes the CH accept temporarily unencrypted pairing requests from the multicast channel. The security flag on the IEEE802.15.4 control field set to zero (see Figure 3.2) indicates no secured frame incoming. Even if in some cases the automatic discovery is desired, it is true that an Out of Band (OOB) auxiliary channel offers a good additional security mechanism in spite of user interaction[29]. Other protocols also use an OOB such as typing a PIN in Bluetooth devices.

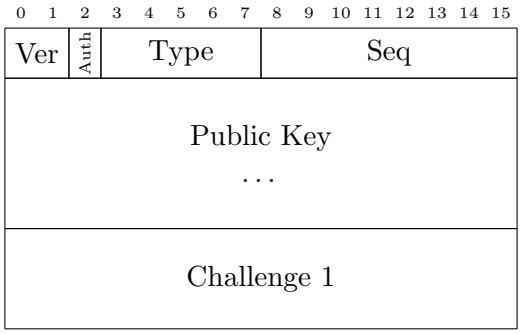


Figure 5.4: *Pairing request message*

After pressing the pairing button for a few seconds on both devices, the newcomer sends a *Pairing Request* message (Figure 5.4) to the multicast channel. This message contains the node's public key (Q_A, computed over the elliptic curve K-163) and a challenge (*Challenge 1*).

5.2.2 Computing the Session Key

When the CH receives this message, it has to compute a session key, which consists of the common point on the curve using ECDH as in Equation (4.10) and *Challenge 1* and *Challenge 2* appended. With this information the CH generates a new session key via KDF2 as in Equation (4.20).

The use of the challenges to generate the key makes sure that if two node must renew a

Field	Length (bits)	Value
Authentication	1	1
Type	5	`PAIRING_REQUEST`
Sequence Number	8	$seq = rand()$
Public Key	536	The public key of the node in format as an elliptic curve point (x,y,z). The curve is 163 bits, therefore each coordinate is 11 words (22 bytes) and one extra byte as a normalization flag (Q_A)
Challenge 1	32	Challenge to provide network authentication and to add randomness to the link key

Table 5.2: *Pairing request message description*

link key for any reason, a new key will be used every time despite not renovating their key pair.

Finally, the CH replies with a *Pairing Response* message (Figure 5.5). This message contains its own public key (Q_{CH}), a signature of the message using the just computed session key, *Challenge 2* and a group key disclosure time in seconds. In Table 5.3 the response message fields are detailed.

The sequence number is increased in every step in order to provide integrity and data freshness to avoid reply attacks, which can be detected thanks to the signature.

The signature is computed as described in Section 4.8, by hashing the message, encrypting it with the newly generated session key and encrypting it using AES-128. The signature is a 4 bytes truncated value.

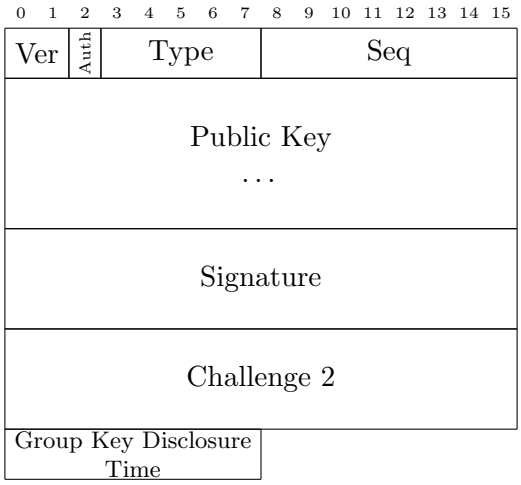


Figure 5.5: *Pairing response message*

5.2.3 Pairing verification

When receiving this response message, Node A computes the session key in the same way CH did and verifies the signature of the message detecting possible contradictions. Finally, generates a *Pairing Proof* message. This message is a signed empty with an increased sequence number. It is detailed in Figure 5.6 and Table 5.4.

Field	Length (bits)	Value
Authentication	1	1
Type	5	`PAIRING_RESPONSE`
Sequence Number	8	$seq_{request} + 1$
Public Key	536	Q_{CH}
Signature	32	Signature of the packet using the just computed key. It is used for verification and weak authentication
Challenge 2	32	Random challenge to provide strong authentication and to add randomness to the link key
Group Key Disclosure Time	8	Seconds before group key disclosure

Table 5.3: *Pairing request message description*

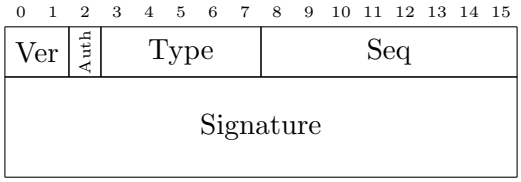


Figure 5.6: *Pairing proof message*

At this point both nodes have a session key which will not expire, therefore it has to be kept secret. The reason for the session key not expiring is fully detailed in Section 5.5. However, this protocol contemplates optional on-demand key renewal, offering a new layer of security upon request of the nodes.

The newcomer is not yet fully authenticated, and it will receive the group key making it full member of the network after the time mentioned in the *Group Key Disclosure Time*. Meanwhile, CH will trigger an authentication request process with the TAS (Section 5.3). The node must contact the CH back after the time indicated in the field *Group Key Disclosure Time* of the *Pairing Response* message. In this message Node A provides the solution to *Challenge 2* to the CH which will validate it and will provide back the solution to *Challenge 1* and the group key information requested.

This procedure is detailed in the following sections Authentication and Authorization (5.3) and Group Key Distribution (5.4). The complete protocol is represented in Figure 5.7, where the messages and operations are specified.

5.3 Authentication and Authorization

After the first pairing, a secure channel is set up between the two nodes. This means that they know the session key and they are able to communicate in a encrypted fashion using AES-CCM (see Section 3.3). Weak authentication is guaranteed thanks to ECDH, albeit the newcomer is still not authorized neither authenticated.

It is one of the purposes of the described protocol to ensure strong authentication between the node and the network at the same time as authorization. This guarantees that the node is the one that it claims to be and that has the right to use the network or the cluster and the network is a genuine one. This is purpose of the TAS.

Field	Length (bits)	Value
Authentication	1	1
Type	5	`PAIRING_PROOF`
Sequence Number	8	$seq_{response} + 1$
Signature	32	Signature of the packet using the just computed key. It is used for verification

Table 5.4: *Pairing proof message description*

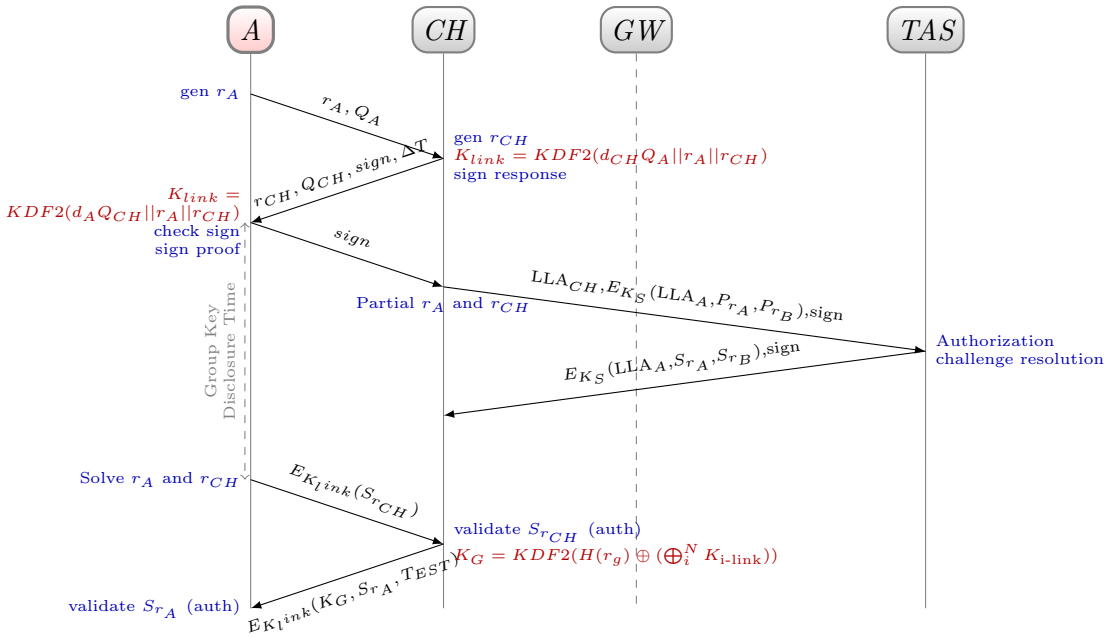


Figure 5.7: *Single node join. Complete protocol*

CH contacts the TAS requesting authentication and authorization of the newcomer by sending a *Authentication Request* message. The message contains the CH and the newcomer's LLAs as well as the partial solutions of the two challenges as it is reproduced in Equation (5.1). LLA_A and the partial solutions of the challenges are encrypted using the CH secret key to ensure end-to-end privacy. Additionally, this message is signed using the private key of CH which it only shares with the TAS to prove authenticity and integrity of the request, and therefore the authorization to request that information.

$$Partial = T(H(E_{\text{link}}(H(challenge)))) \tag{5.1}$$

The TAS validates the message signature using LLA_{CH} secret key and decrypts LLA_A and the partial solutions. If the Node A is authorized the TAS computes the final solution to the challenges (1 and 2) using Node A secret key and responds to the CH with an *Authentication Response* message, the message is signed to authenticate the origin and, as in the request message, the LLA_A and the solutions are encrypted using CH secret key for end-to-end security. In case of not authorization, the solution of the challenges is not present in the message and the CH will automatically unpair with Node A.

It has to be noted that the message sensitive data is encrypted end-to-end as it is important that no middle node collects important data about the challenge resolution of one of the nodes.

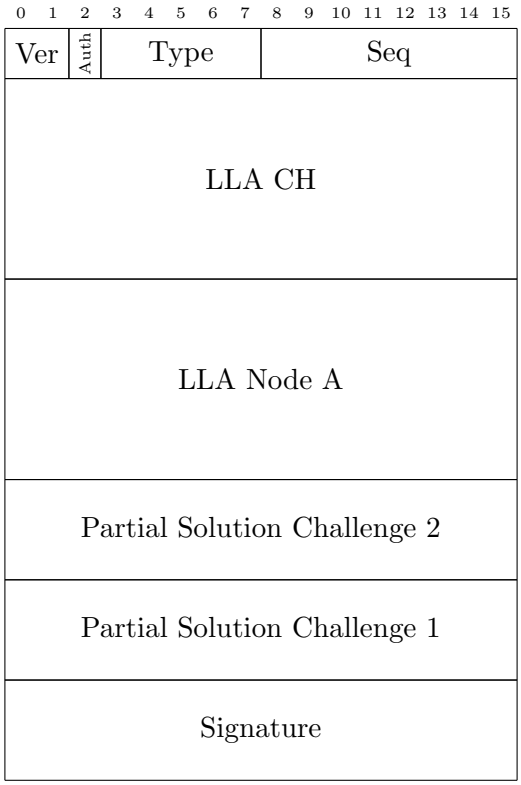


Figure 5.8: *Authentication request message*

Field	Length (bits)	Value
LLA CH	64	Link-local Address of CH
LLA Node A	64	Link-local Address of Node A
Partial Solution Challenge 2	32	Challenge proposed to Node A by CH
Partial Solution Challenge 1	32	Challenge proposed to CH by Node A
Signature	32	Signature of the message using the secret key of CH

Table 5.5: *Authentication request message description*

Although trust is assumed among the nodes of the network but it may be the situation of a not detected intruder within the route.

After the *Group Key Disclosure Time* Node A will request a new key via a *Group Key Request* message to CH sending the solution to *Challenge 2*, if the solution is correct (matches the provided by the TAS) the CH will generate a new group key, and send it together with the solution to *Challenge 1* to Node A. Then it will distribute the new key among the rest of the nodes.

5.3.1 Authentication via Web Server

Due to the nature of Tado's network and products and for scalability reasons, the TAS consist in a cloud web server, therefore all the requests must be made over HTTP protocol. Thus, every connection turns into a request-response pair. Additionally, it is a requirement that the user must approve manually a device joining to the home network.

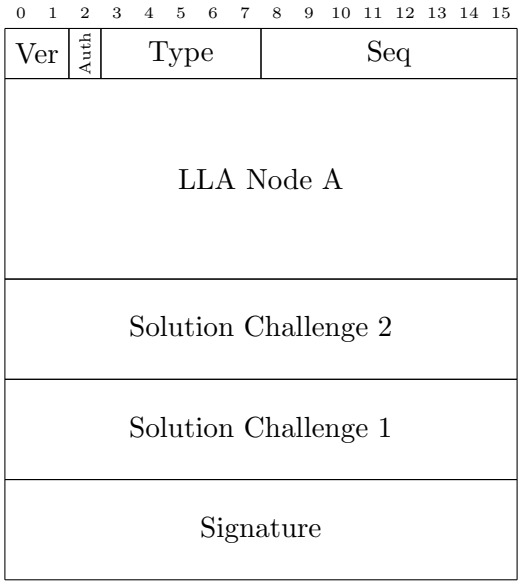


Figure 5.9: *Authentication response message*

Field	Length (bits)	Value
LLA Node A	64	Link-local Address of Node A
Solution Challenge 2	32	Challenge proposed to Node A by CH
Solution Challenge 1	32	Challenge proposed to CH by Node A
Signature	32	Signature of the message using CH secret key

Table 5.6: *Authentication response message description*

An adaption layer to the protocol has been developed. Immediately after a *Pair Proof* message has been received, the CH communicates the joining to the TAS using an HTTP request containing as binary payload the *Join Notification* message as represented in Figure 5.10. The first field, the LLA_{CH}, and the rest of the message is encrypted using AES-CCM-128 with the first 4 bytes of the message as a nonce with the CH secret key as the encryption key. This way, integrity, confidentiality and authenticity are ensured at the same time as protecting from reply attacks.

The reply of the server is typically a 200 OK if the requested device is under consideration, a 404 (not found) if the device is not known and a 401 (unauthorized) if the device should not be considered anymore, following the standard HTTP codes[31]. When the device is under consideration, the user receives a notification on the dashboard and his/her smartphone to approve (authorize) the device.

After a fixed amount of time which is set to 3 minutes[1], the CH sends another binary-embedded HTTP request to the server, this time a full *Authentication Request* encrypted in the same way as in the previous message using AES-CCM-128. The server will reply with an HTTP response containing the binary encrypted *Authentication Response* message in case it is authorized or with a 401 status code otherwise. In case that the user did not reply to the authorization request in this interval of time, the node will be considered unauthorized.

This adaption allows the integration of a web server as a TAS and provides the user

[1]This time is experimentally found during the work time on the thesis, it may not be the same as it will be influenced by users feedback and further usability cases for Tado.

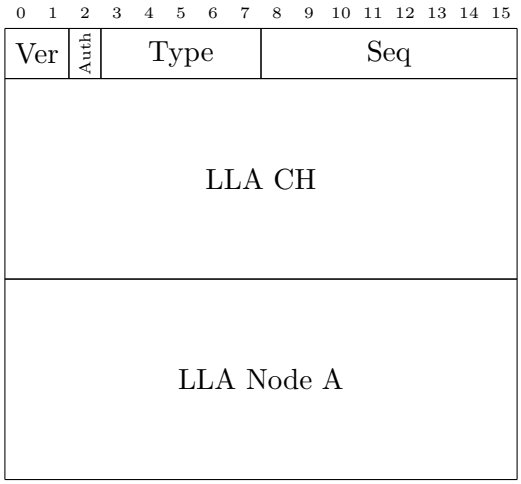


Figure 5.10: *Joining notification message*

the control of the devices accessing their home network in a completely transparent way for the rest of the network and at the same time ensuring the same level of security with just an extra message. Furthermore, it allows high scalability in spite of some overhead but only most powerful nodes such as CH or GW can perform such TCP communications and therefore new nodes must be firstly paired to those devices.

The complete procedure can be seen in Figure 5.11. In this figure, a node A joins the network using the CH, the TAS is a webserver and user/customer interaction is required for authorization as described above.

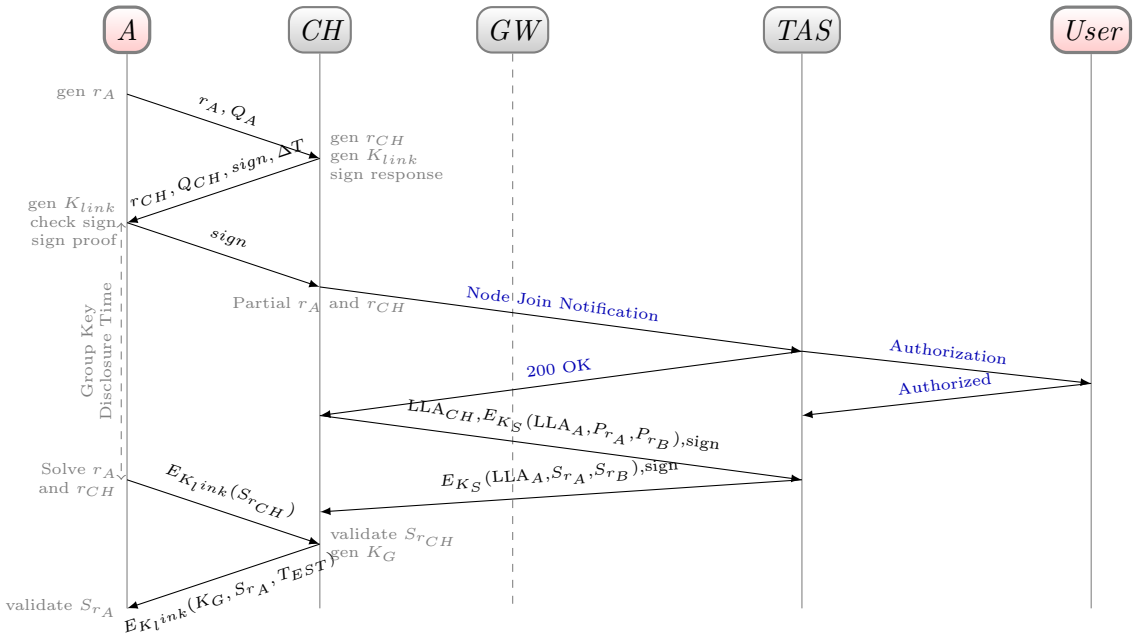


Figure 5.11: *Single node join. Complete protocol using webserver as a TAS and customer interaction*

5.4 Group Key Distribution

The group key distribution may happen in three situations: when a node joins the network, leaves or is revokated.

5.4.1 Node Join

After the Group Key Disclosure Time, Node A will send a *Group Request* message to CH with the authentication flag set and the solution to *Challenge 2*. If the solution of the challenge is correct, CH will compute a new group key (see Section 4.6.4) and send it to Node A together with the solution to *Challenge 1* which was proposed by the newcomer, and the *Key Establishment Time*. This is the number of seconds the node should wait until establishing the new group key. The value of it is interesting as it depends on the size of the network and if the nodes have current operations pending, they can delay the key establishment until the tasks are finished. Finally, the CH distributes the key among its neighbours using the same *Group Key Response* message.

If the solution to the *Challenge 1* is correct, Node A will set the new key after the time indicated in the establishment field. Otherwise the entry on the neighbour table for the CH is voided.

The *Group Request* message and *Group Response* message are represented in Figure 5.12 and 5.13 and in Table 5.7 and 5.8.

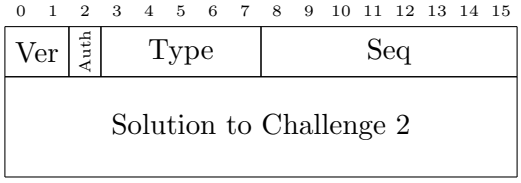


Figure 5.12: *Group request message*

Field	Length (bits)	Value
Authentication	1	This flag is set to one whenever a node has not been authenticated by the network yet (or the other way round)
Type	5	`GROUP_REQUEST`
Sequence Number	8	$seq_{pair\ proof} + 1$ for authentication or $seq = rand()$ for regular request
Solution to Challenge 2	32	The solution to the previously proposed challenge by the network

Table 5.7: *Group request message description*

5.4.2 Existent Node

When an existent node receives a *Group Key Response* message it automatically sends a *Group Key Advertisement* to the multicast channel. When a neighbouring node receives this message and has not received the new group key, it will send a *Group Key Request* message to

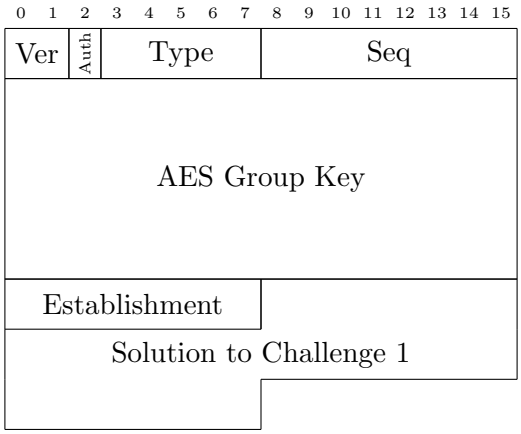


Figure 5.13: *Group response message*

Field	Length (bits)	Value
Authentication	1	This flag is set to one whenever a node has not been authenticated by the network yet (or the other way round)
Type	5	`GROUP_RESPONSE`
Sequence Number	8	$seq_{request} + 1$
AES Group Key	128	Group Key
Establishment	8	Number of seconds to wait before key set up.
Solution to Challenge 1	32	The solution to the previously proposed challenge by the node

Table 5.8: *Group response message description*

the advertiser. This procedure is repeated for all the nodes within the network. This is done because the broadcast messages are less expensive than the directed messages and because a node may be neighbour of two nodes knowing the key, which makes it cheaper in terms of energy that the interested node requests the key to one of the knowing nodes.

The *Group Key Advertisement* message guarantees that the CH does not have to send the message to all the nodes individually, distributing the load of the network and using link level key distribution rather than entire routes, hence reducing the communication overhead of the nodes near the CH.

The challenges in the messages are not applying any longer and therefore they are not sent if the authentication flag is not set. Thus, this is reducing the size of the messages. Furthermore, the sequence number of the request message must be the incremental to the pair proof for the authentication case and a random number for the advertisement case. It should not be the incremental in the latter case to avoid encrypting the same message with different encryption keys, which is a known technique of cryptoanalysis. This would not be dangerous as the encryption takes into account the sequence number of the IEEE802.15.4 frame, however it is considered a good practice to use *semantic encryption.*

5.4.3 Node Revocation

When a node must be revoked the CH revokes the node first and generates a new group key. Then it distributes a *Revoke* message among its neighbours individually. When a *Revoke* message is received and the node has a neighbour responding LLA given, this neighbour is automatically revoked. The message is forwarded individually to all the neighbours but the affected one. This message follows the structure as in Figure 5.14 and contains the new group key to be set with immediate effect as described in Table 5.9.

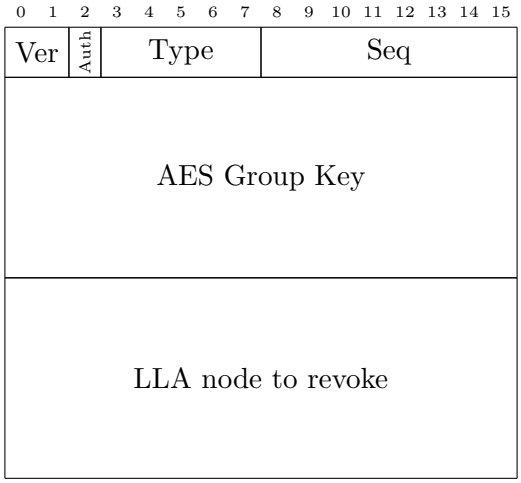


Figure 5.14: *Revoke message structure*

Field	Length (bits)	Value
Authentication	1	0
Type	5	`REVOKE`
Sequence Number	8	*rand*()
AES Group Key	128	New group key
LLA of node	64	LLA of the node to revoke

Table 5.9: *Revoke message description*

5.5 Key Renewal

There are four types of keys in the proposed solution for the Key Generation and Distribution Problem. The keys scopes are detailed in Section 4.6. The secret key and the key pair are never renewed.

Recent research shows that AES-128 as in FIPS 197 (10 rounds) is vulnerable to brute force attacks with a complexity of $2^{126.1}$ instead of the expected 2^{128}[10]. Considering that the smallest frame of the protocol is 26 bytes (Table 6.5), and that the IEEE802.15.4 standard (see Section 3.1) forces a data throughput of 50kbps at 868MHz, a complete brute force attack under this conditions requires $2^{93.2} \approx 1.11 \cdot 10^{28}$ years to exhaust all possibilities considering no processing time and no collisions in the medium. This makes brute force not a good option to recover an AES 128-bit key and enforces the minimum or even no key renewal.

5.5.1 Link Key

A link key renewal can be triggered by one of the nodes when it considers it necessary. Given the throughput of the communication, the strength of the cipher and the limited resources of the nodes, it is not advised to renew the link key. However, the structure of the protocol allows key renewal in an easy way and without mediation of the TAS.

When a node considers that a link key must be renewed –usually because of the statistics of messages sent through a link– this sends a *Pairing Request* message to the other end, in an encrypted way as the nodes are already paired, with the authentication flag set to 0. The process will continue as the Pairing process (Section 5.2), without proceeding to authentication.

If a flaw in AES should be found (e.g. lowering the time required to break a key), the protocol allows optional key renewal. Additionally, the version field allows a new version of the protocol to be released. However, the use of the version flag must offer retro-compatibility to make it attractive for commercial solutions.

5.5.2 Group key

The Group key can only be renewed by the CH when a node joins the network, leaves or has been detected as compromised.

The CH sends the group key to its neighbours individually. When a node receives a new group key, it sends a *Group Key Advertisement* message to the multicast channel; if neighbouring nodes need to renew a group key they will send a *Group Key Request* message to that node, which will reply with a *Group Key Response* containing the new group key.

Compromised Node

If the group key has to be renewed because of a node revocation, the CH will use a *Node Revocation* message (Section 5.6) sent individually to all its neighbours except the affected node. It has immediate effect, unlikely the regular procedure when the key establishment can be delayed.

5.6 Revocation

The described solution also contemplates the case of a node access revocation. The special packet will void all the possible communication of the suspicious node with its neighbours and thus with the network. The TAS sends this message to the area CH, which distributes it among its neighbours as described in Section 5.4.3. The revoked node will be left out of the network.

No black list of nodes is kept in the memory of the nodes. Firstly, because of memory overhead it may produce and secondly, because in case of a protocol or TAS error a node may become unusable. If a network may consider by mistake a node to be malicious, a new message structure to unblacklist the node must be implemented, which may introduce new security flaws. The user experience and the possibility to re-pair a node is specially important commercial applications, hence the solution must always allow manual pairings. Since to trigger a manual pairing the pairing button of both parties must be pressed during N seconds, it makes continuous pairing attempts not dangerous in terms of a possible Denial of Service (DoS) attack as every retry is too time expensive.

5.6.1 Node Re-pair

Another function of the revocation messages is to allow a defect node to re-pair. In the case of a node which looses the keys to its neighbours, it is isolated from the network. Even after a new manual pairing with the CH, the automatic pairing process (Section 5.7) will not be triggered as the neighbours already have knowledge of the affected node, hence they consider a certain link key. For that reason, when a node re-pairs to CH, the group key is distributed among the rest of the nodes using a *Revoke* message instead of a *Group Response* message. Nodes delete the information about the affected node and, since no black-lists are kept, the automated pairing can be triggered after the manual pairing. This behaviour is a good approach for consumer products in order to overcome defect firmware where the keying information is deleted by mistake or new firmware version with imposes new key storage schemes.

5.7 Automated pairing

When a node is member of a network, this means that it has the group key and it can process multicast messages such as Neighbour Solicitation (NS), RA, ... It is able to pair automatically with other unknown devices. This is done in the same fashion as the described pairing procedure. First, the node sends a message to the multicast with an *Pair Request* message, however the authentication flag is not set.

A node receiving a pairing request (Node B) from an unknown node (Node A) sends an *Authenticity Request* message to the multicast. If another node (Node C) is successfully paired to the requested device it means that Node A is truly authenticated and Node C will reply to Node B with an *Authenticity Confirm* message. When at least two responses have been received, Node B responds to the unknown node's *Pairing Request*.

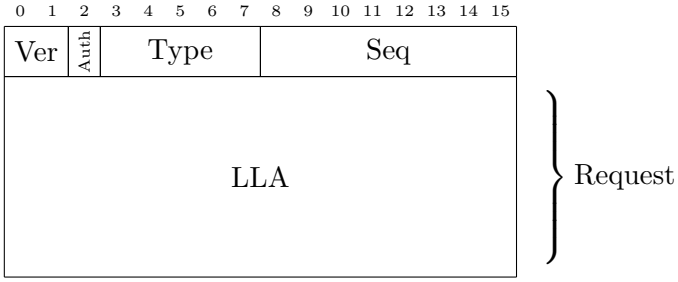


Figure 5.15: *Authenticity messages*

Field	Length (bits)	Value
Type	5	`AUTHENTICITY_REQUEST` or `AUTHENTICITY_CONFIRM`
Sequence Number	8	$seq = rand()$
LLA	64	Requested node LLA

Table 5.10: *Authenticity messages description*

In case no node replied to the *Authenticity Confirm* message, Node B sends this message to the CH to confirm authenticity.

The two responses are required because in case of one node is compromised, both the group key and the link key can be read out, and therefore the attacker may be able to reply to the authenticity message. In that case, it is considered that two nodes have to be compromised without detection before confirming the authenticity of a third node. This is a more unlikely situation. Both nodes must be neighbours of Node B and both attacks must have not been detected.

The *Authenticity Request* and *Authenticity Confirm* messages are reproduced in Figure 5.15 and Table 5.10

As described before, the messages responses must have the incremented sequence number as the requested one.

Chapter 6

Evaluation of the Proposed Scheme

6.1 Security Achievements

In this section the security achievements by the proposed solution are analysed. Firstly, the different attackers are defined and then the individual contexts are discussed depicting these attackers and the harm threat in the situation.

6.1.1 Types of Attackers

An attacker is considered an *insider* as she has already compromised some authorized nodes in the network (run malicious code, stolen keys, ...)[48] and an *outsider* when she has not special access to it. Attackers can be distinguished as well according to their behaviour and resources.

Attacker behaviour

An attacker is considered an *oblivious attacker* when she may select the next node to attack randomly, and the *smart attacker*[13, 56] when she will select the next node which is most likely to provide significant information. The smart attacker is very dangerous when the schema is key pool pre-loading, as it can lead to a key exhaustion.

Attacker Resources

In function of the resources available by the attacker we can distinguish two types of attackers[37, 48]. The *Mote-Class* attacker has access to nodes with similar capabilities, in opposite to the *Laptop-Class* attacker who owns more powerful devices such as a laptop with higher processing capacity, battery capacity and transmitting power. The last can easily cause energy exhaustion (DoS) of the nodes, being able to isolate sectors of the network.

6.1.2 Adversaries

In this sections, several attacks and vulnerability contexts such as *fraudulent device*, *fraudulent network*, *replay attacks* and, of course *tampering*, *eavesdropping* and *MiM* are analysed.

Eavesdropper

An eavesdropper can listen to the pairing communication, however, thanks to ECDH it will not be able to deduce the shared key and therefore the proposed solution is not vulnerable at any moment to eavesdroppers. When the devices are paired, all the communication is encrypted, hence not exposed to eavesdropping.

Fraudulent Device

A device is considered fraudulent when it attempts to fake the identity of a device, but this will not be allowed by the network. The challenges and their resolution proposed in this thesis protects the network from fraudulent devices and Man-in-the-Middle attacks as they provide strong authentication.

Tado TAS has logic to detect possible attacks of a fraudulent node, e.g. same node is trying to connect to different networks or does reiterated challenge solution requests. Thus, thanks to the double authorization layer in Tado it is possible to block joining requests from a particular device automatically or by the customer.

In order to choose the optimal size for the challenges, the Equation (6.1) was used. This equation describes the maximum time a brute force attack needs to be successful. T_{button} is the time a pairing button has to be pressed in order to start the pairing process; $l_{p,x}$ is the length of a *pairing* message where x is the request, response or proof messages. $\overline{R_x}$ is the average number of retransmissions due to *collisions* and due to the *slot* of the RDC; $T_{g,disc}$ is the time after which the network will provide the group key, and with it the authentication and authorization confirmations. T_{comp} is the key computation time. Finally, b is the size of the challenge in bits.

$$t_{brute\ force} = (2^b - 1)\cdot \left(T_{button} + \frac{l_{p,req} + l_{p,resp} + l_{p,prov} + l_{g,req}}{v} \cdot (\overline{R_{collision}} + \overline{R_{slot}}) + T_{g,disc} + 2T_{comp}\right) \quad (6.1)$$

$$t_{brute\ force} \approx (T_{button} + T_{g,disc} + T_{comp}) \cdot (2^b - 1) \quad (6.2)$$

In the worst case, *laptop-class* attacker, the $2T_{comp}$ turns into T_{comp} as the computation time is almost null. This equation is plotted in Figure 6.1 (red) where the amount of time necessary to perform brute force attacks on challenges of 8, 16, 24 and 32 bits was marked.

The ideal case is considered in (6.2), no retransmissions ($(\overline{R_{collision}} + \overline{R_{slot}}) = 1$), no key computation time ($T_{comp} = 0$) for the fraudulent device and a MSP430 computation time for the network ($T_{comp} = 1.986s$, Table 6.3). The $T_{g,disc}$ is typically 3 minutes and the button pressing T_{button} is typically 5 seconds, which leads to $t_{brute\ force} \approx 25.000$ years for a 32-bit challenge. Hence, brute force attack is not likely to be successful.

In the situation of an attacker triggering the *automated pairing* procedure, the node contacts the CH since the neighbouring nodes do not reply to the *authenticity request* message. The CH is able to detect the attack and therefore revoke the fraudulent node in case necessary. Two neighbouring nodes must be tampered at the same time when the automated pairing process is triggered, leading to a more unlikely situation.

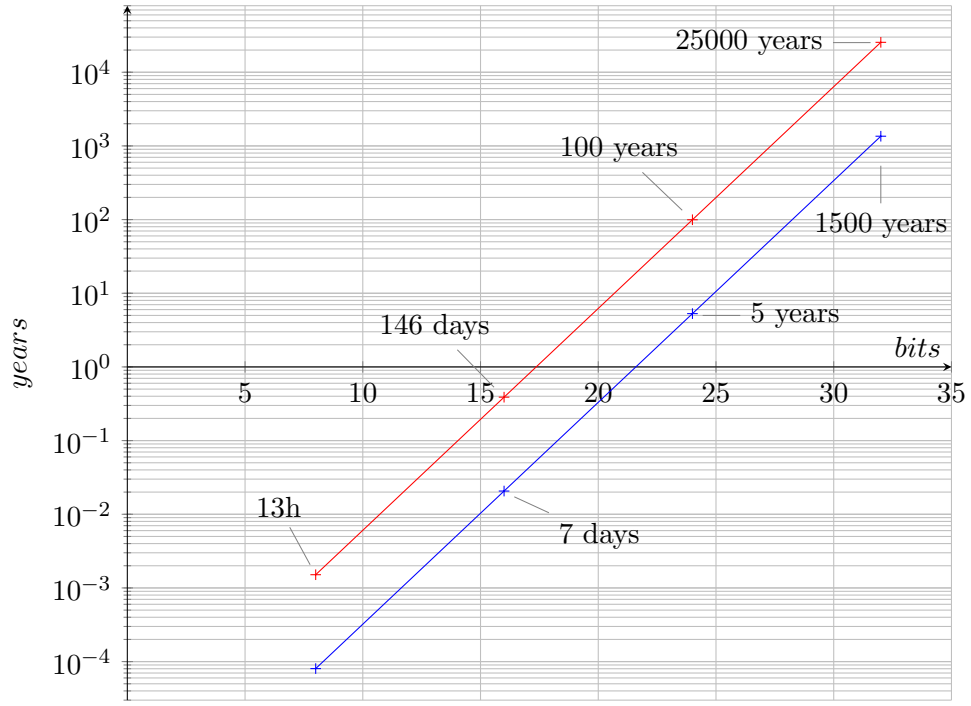


Figure 6.1: *Challenge break brute force attack duration versus challenge length. The red curve represents a fraudulent device connecting to a valid network, while the blue curve represents a genuine device connecting to a dishonest network*

Fraudulent TAS or Network

A fraudulent network or even the TAS can attempt to pair a device; in that case, the challenge proposed by the device will fail to be solved. Even considering no retransmissions, and no key computation by the network size, the node still has to compute a key, signatures, etc. taking $4.916s$ for the CC430. Equation (6.3) indicates the most influential terms, which are represented in Figure 6.1.

$$t_{brute\ force} \approx (T_{button} + T_{comp}) \cdot (2^b - 1) \tag{6.3}$$

Message Replayer

Message replaying is a really harmful attack on a wireless medium. However, the capabilities of the proposed solution make it not vulnerable to this kind of attacks.

During the pairing bootstrap there are three sequence numbers transmitted in clear text, IEEE802.15.4 has a 1 byte of sequence number, 6loWPAN UDP has 1 more byte of sequence number and the proposed protocol also has 1 more byte of sequence number. Since the key exchange is performed in the application layer this application sequence number is required in order to be able to sign the messages and ensure integrity and authentication. The goal of this sequence number is to avoid message reply attacks. It is incremented on each message, hence the ends know the expected sequence number during pairing, not accepting other messages. Thanks to the signature it can be ensured that it is not a replayed message even if lower layers are trespassed.

Physical attack (tampering)

The solution is in some measure vulnerable to tampering attacks. These are difficult to solve while ensuring strong authentication capabilities. Once a device is tampered, it is considered that all its data is disclosed.

In the case of a tampered node and the group key read out before being able to change it or detect the intrusion, an attacker may try to pair the devices with neighbouring nodes. This is not possible in the proposed solution as for a new automated pairing, the nodes will request authentication information of the newcomer from their authorized neighbours. In case of an unknown node, the CH is queried. A node must tamper two nodes which are neighbours to a third one in order to be able to access the network with a false newcomer. Albeit being an unlikely situation, an *smart attacker* could trigger that situation. Only an IDS can detect this breach.

When a node has been tampered and its secret key is extracted, a fraudulent device may attempt to pair using this information. Some [36, 53, 54] have considered that devices cannot be attacked during bootstrap and after the initial pairing they can delete the secret key. Hence, tampered nodes will not reveal secret information which may grant access to the network. Since keys are never renewed, this is a good approach except in the case of a memory-loss a node, in which the node may become unusable. This situation is not convenient for commercial products, and therefore it is encouraged to relay on a IDS. The TAS however can detect this attacks easily and react to them. A node trying to pair within to different clusters is an unlikely situation as well as several pairings of a node in the same cluster within a short period of time.

On the other hand, when a device is captured, not only the neighbouring links are compromised but the group key is disclosed as well. So that, an attacker can decrypt the broadcast messages. Performing periodical authentication may not provide of trustworthy results as the secret key has been disclosed. Again, only a IDS may help, alerting the TAS that a device has been compromised, and therefore voiding all its pairings and blocking it not to authenticate again.

The worse situation is a tampered CH. It is impossible to detect if it has no interaction with the TAS. A non-tamperable CH has been considered by many[26, 34, 43, 50], but this is not an ideal situation. In Tado, a periodic connection from the CH to the TAS is used to report health of the network and other information. In case of disruptions on this channel, a possible attack may have occurred. Redundant CHs are recommended by others[12] to solve the mentioned problem for middle-sized clusters.

Man-in-the-Middle

MiM attacks are dangerous during the communication bootstrap when the first key exchange is made. If an attacker is able to forge the LLA address of the device and making the network consider the messages coming from this address, it can be paired to the network in behalf of the other node. In that case, the node and the network are not going to authenticate each other in this schema thanks to the partial solution of the challenges where the link key is involved in the solution and this is generated through ECDH. This situation is represented in Figure 6.2.

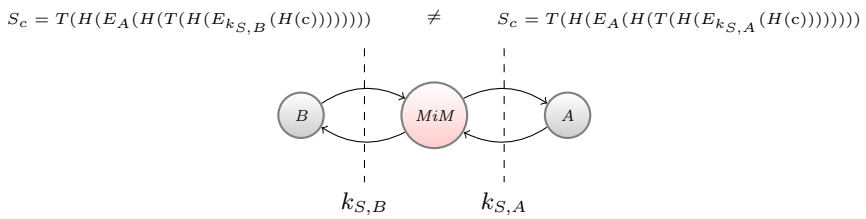


Figure 6.2: *Man-in-the-middle scenario. A different key for each link is used thanks to ECDH*

6.2 Memory Overhead

In order to fit Contiki, Relic and the key generation and exchange layers, several modifications were done in Contiki's source, in the CC430 drivers and in Relic Toolkit source code. However, the numbers presented in this section shall not be considered absolute but merely illustrative as they depend on the software implementation and compiler optimizations. Nevertheless, to accommodate ECC in embedded devices together with other applications is a challenge hence, the difficulty of the task shall be remarked. The variance of the sizes depending on the compiler and platform can be observed in Table 4.2 where a variance of about 3KB must be noticed.

6.2.1 Tool Chain

All the provided data for the CC430 and MSP430 is result of compiling with `msp430-gcc` dev 20120911 for `gcc` version 4.7.0 20120322. The binary sizes are obtained using the `msp430-size` and `msp430-nm` for the symbols sizes, both of the same development version as `msp430-gcc` and belonging to the tool set GNU Binutils 2.22.

Regarding the Cortex M3 gcc version 4.5.1 (Sourcery G++ Lite 2010.09-51) was used as a compiler. And nm and size from GNU Binutils 2.20.51.20100809.

The compiler optimizations have been *-Os -fno-strict-aliasing -ffunction-sections -fdata-sections* and *-Wl,–gc-sections* for the linker to strip not used sections for both compilers, leading to a reduction of the binary size, in spite of a diminution of the speed.

6.2.2 Contiki

As mentioned previously, Contiki OS (Section 3.4) over the platforms specified in Section 3.5 was used for the implementation of the proposed solution. For being able to integrate Contiki OS in the smallest platform together with the cryptographic library, some important modifications in Contiki were carried out. This modifications were such as: rewriting memory management functions, rewriting some code of the rime module, blocking the response and processing to neighbour solicitations and advertisements by other nodes, disabling TCP and some driver structures, reformat the default driver structure to behave nameless and write custom printf functions for debugging via UART among others. After the trial of several configurations, the size of Contiki was trimmed from 24KB to 18KB. Taking into account the 32KB of flash memory on the CC430, it is a significant improvement[1].

[1]The modifications do not modify the behaviour of Contiki but improve its speed and reduce the code footprint. They will be committed to the Contiki OS source project in Github.

6.2.3 CC430

Important modifications were required for this platform. Drivers to use AES and CRC peripherals were written during the process in order to avoid using the heavier and slower software implementations, thus reducing the processing time as well as the memory footprint. This improvements can be seen in Table 6.1. Which makes a significant difference of 2.6KB.

Module	**Software**	**Hardware**
AES encryption	1040 bytes	116 bytes
AES decryption	1350 bytes	98 bytes
AES configure key	n/a	46 bytes
AES SBOX	512 bytes	0 bytes
AES-CCM encryption	32.349 ms	5.789 ms
AES-CCM decryption	32.349 ms	5.798 ms
CRC16 (CCITT)	52 bytes	24 bytes
Total	2954 bytes	284 bytes

Table 6.1: *Comparison between software and hardware implementations sizes of AES and CRC-(CCITT)*

The memory footprint reduction is truly significant and critical for this microprocessor. Additionally, the AES encryption and decryption speed is also reduced perceptibly leading to a reduction of the power consumption.

6.2.4 Relic

Relic-Toolkit is meant to be highly configurable and modular. It can be compiled as an external library and linked afterwards in the final binary. The optimizations for lower memory footprint were described in section 4.3.5. Since the secret pair keys are not meant to be renewed, there is not need to include key pair generation algorithms in the devices, and therefore the final memory footprint is: 4.34KB for the CC430 and 3.33KB for the Cortex M3.

6.2.5 Key Manager

The key manager is the application in charge of key exchange, node authentication and all the operations related to the key generation and distribution. It has two parts, one is the application, which manages the key renewal and processing, and the other is a low level layer which encrypts and decrypts packets when they are received or transmitted.

It has to be noted that in this case the memory overhead is the most variable factor as it will depend on the implementation which is at the time of writing this thesis, the very first version. The total size of the implementation of the complete protocol on Contiki OS is 2160 bytes for the CC430. This figure can not be considered absolute as compiler optimizations may have merged low level UDP functions into the sending and receiving procedures[2] Although it is a large number compared to the size of Relic-Toolkit, it must be considered that the complete protocol, considering retransmissions and packet flow, is accounted.

[2]Additionally, all the cryptographic functions have been trimmed off during the computation.

6.2.6 Infomem

Infomem is a region of the flash memory in the microprocessors where the page size is inferior and therefore it is ideal to store permanently variable information during runtime. MSP430 and CC430 offer a infomem section of 512bytes (4 pages of 128 bytes); therefore this is the maximum amount of key-address mapping that can be stored in the sensor. Taking into account that an address is 8 bytes and an AES key is 16 bytes, this makes a total of 20 neighbours considering that the group key is also stored in the infomem. Thus, all the simulations in later sections consider 20 neighbours as a top boundary for large sized networks. It should be remarked that this is not realistic for most networks as in this section usually the serial number and local information such as local RF address is stored taking about one entire page for this information.

Other approaches may consider the use of an entire flash page out of the infomem section to store this information, hence more data can be stored as the page size is larger. However, the redundancy of the data is critical for both approaches, as a power failure while writing the segment or an interruption can cause the data to be corrupt and therefore the node to have untrustworthy security information. This is not in the scope of the analysis in this thesis but must be considered for implementation being a critical factor in commercial products which may lead to user dissatisfaction due to unusability. This is especially critical in the second case as the pages must be written at once and therefore all the information in the whole page may become corrupted.

6.3 Processing Overhead

6.3.1 Test Set-up

The set-up consists in a 10Ω resistor connected between the ground pin of the chipsets and the real ground (Figure 6.3). This allows to compute the current flow through the resistor and offers more accurate measurements than reading on the Vcc port in the available testing devices.

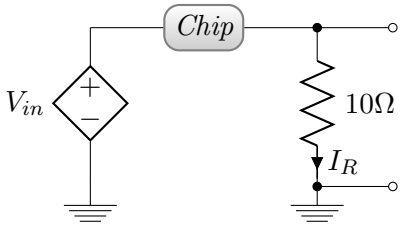


Figure 6.3: *Measurement circuit*

Using the described procedure, measurements of the energy consumption when the processor is busy and when the CC1101 (or CC430) is transmitting and receiving were carried out. The time consumption of the different operations required by this proposal are computed in Central Processing Unit (CPU) cycles and converted to seconds. Using this information, the energy consumption of the devices is computed.

The measurements for MSP430 and CC430 processors have been done using a test board. A test board has no electronic components other than the strictly necessary for the microprocessor to run, which avoids external energy consumption by the peripherals. The boards can be observed in Figure 6.5 and 6.4. The measurements for the CC1101 chipset

have been done connecting this chip to the MSP430 board and measuring only the intensity drained by this chip. For the radio frequency component on the CC430 chipset, this isolation is not possible since the RF chipset is embedded in the microprocessor. For the Cortex M3 microprocessor tests, a modified Tado gateway board has been used (Figure 3.22). All the peripherals on it have been disconnected in exception of the microprocessor and mandatory electronic circuitry for it to run.

All energy consumption tests have been done with the Rigol DS1052E oscilloscope. The accuracy is described in Equation (6.4) according to its datasheet[52].

$$\begin{aligned} reading \pm\ 3\%(reading + vertical\ position) \\ + (1\%\ of\ vertical\ position) + 0.2div + 2mV \end{aligned} \tag{6.4}$$

Therefore, it must be remarked that in the presented results, a level of uncertainty of at least 0.2mA is present. This is not considered as all the measurements contain errors and uncertainty, since it would complicate the calculations and the readability of this document. Nevertheless, it must be mentioned for awareness of the reader.

Figure 6.4: *CC430 test board*

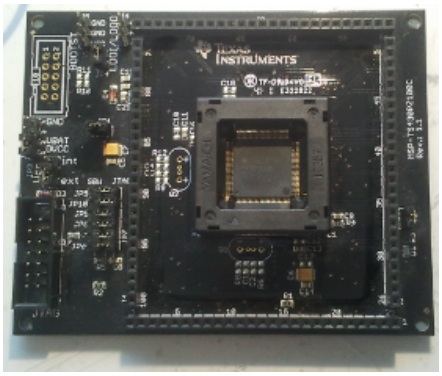


Figure 6.5: *MPS430 test board*

6.3.2 Energy Consumption

The input voltage of the Tado GW mote is always constant. However, it is not the case for the CH and the sensor node, whose input voltage is variable depending on the HVAC output for the previous and on the battery charge and solar cell for the latter. The current consumption has been tested for several input voltages obtaining a lineal dependency as plotted in Figure 6.6. Since these motes can operate at different voltages without influencing the operation frequency, this will affect the energy consumption according to $P = V \cdot I$. Both values –for maximum and minimum input voltage– are given in all the cases. In the case of the CC1101 no current variation was observed when powered at different voltages. Table 6.3 reflects the results. The CC430 shows a similar behaviour with very small variations of the current consumption.

In Table 6.3 the individual processing consumptions for the different chips and algorithms required by the proposed solution are presented, while Table 6.4 holds the summary of consumption depending on the role (newcomer and CH) for the different microprocessors. Two of the most obvious observations are the incredible reduction of processing time thanks to the embedded AES accelerator in the CC430 and the expensiveness of the ECC operations,

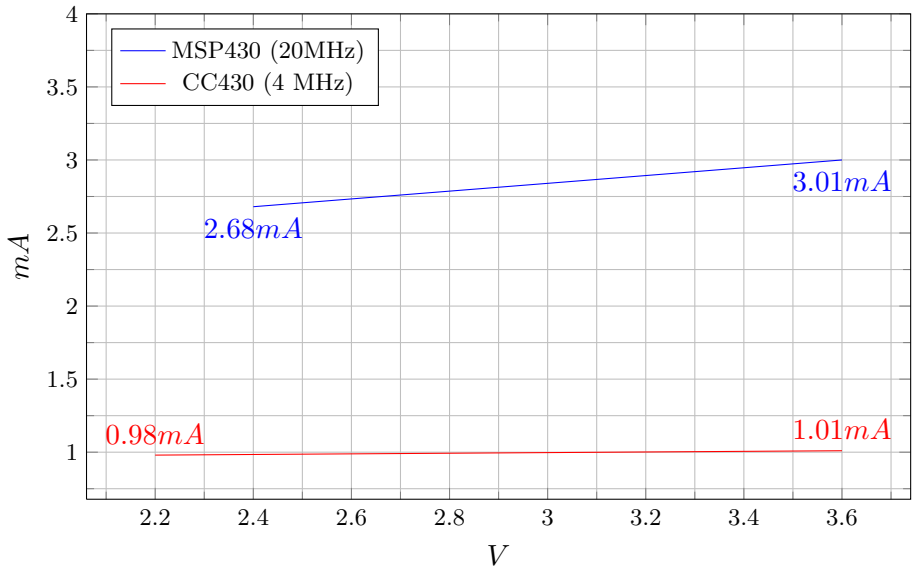


Figure 6.6: *Energy consumption of the different chipsets*

	Input Range	**Minimum** (mA)	**Maximum** (mA)
Cortex M3	3.3 V	43	43
MSP430	2.4 - 3.6V	2.68	3.0
CC430	2.2 - 3.6V	0.98	1.01
CC1101 tx (0dBm)	1.8 - 3.6V	16.8	16.8
CC1101 tx (12dBm)		28	28
CC1101 rx		16.4	16.4
CC430 tx (0dBm)	2.2 - 3.6V	18	18
CC430 tx (10dBm)		32	32
CC430 rx		17.8	17.8

Table 6.2: *Current drain of the different chipsets for their input range at full load*

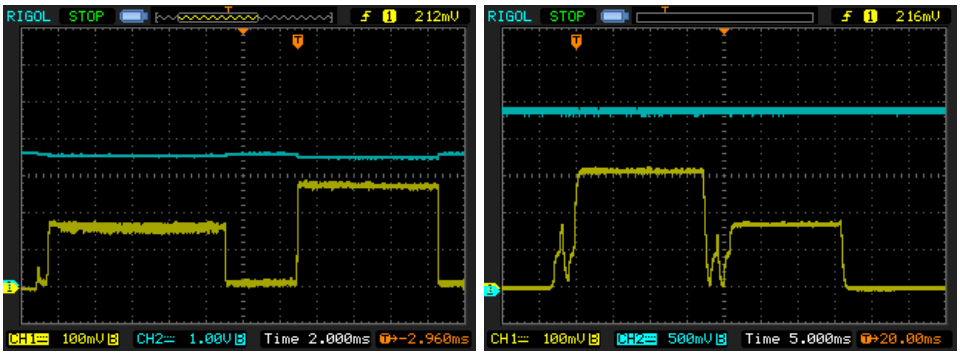
which is the most influential value for the consumption as can be seen in Table 6.4.

As already mentioned, key pair generation is discouraged in the motes, and the energy consumption for these operations confirms the need of avoiding it as much as possible.

According to the described key distribution protocol, a new node joining the network has to solve two challenges for the authentication; to compute a signature for the *Pairing Proof* message and another to validate the *Pairing Response* message. Additionally, it has to derive a key from given point sets. At the same time, the CH has to compute a key derivation for the link, sign a *Pairing Response* message and validate a *Pairing Prove*, it has to compute two partial solutions to the challenges, sign an *Authentication Request* message and finally a derive a new group key. Encryption and decryption is also considered in Table 6.4. The message sizes are extracted from Section Communication Overhead.

It can be observed from the results that the main power consumption contribution comes by the hand of the key derivation function, though the point multiplication algorithm chosen as described in Section 4.3.5 offers a good trade-off between memory overhead and speed. However, faster algorithms may reduce the power consumption. These algorithms are suitable for the MSP430 and the Cortex M3 but were dismissed for the comparison in favour of a

[2]The values presented in Table 6.3 are the result of taking three measurements and computing the average.



(a) *CC1101* **(b)** *CC430*

Figure 6.7: *Oscilloscope readings for full tx power*

Algorithm	Cortex M3		MSP430			CC430		
	time	avg	time	min	max	time	min	max
Hashing	0.665	94 μJ	5.432	35 μJ	59 μJ	15.625	34 μJ	57 μJ
Point Generation	609.792	86.529 mJ	1909.790	12.284 mJ	20.626 mJ	4840.759	10.437 mJ	17.601 mJ
Key Derivation	609.910	85.546 mJ	1912.598	12.302 mJ	20.656 mJ	4845.581	10.447 mJ	17.619 mJ
AES-CCM encryption	5.329	756 μJ	21.606	139 μJ	233 μJ	5.789	12 μJ	21 μJ
AES-CCM decryption	5.313	754 μJ	21.606	139 μJ	233 μJ	5.798	13 μJ	21 μJ
Signature	0.895	127 μJ	6.836	44 μJ	74 μJ	10.925	24 μJ	40 μJ
Challenge Solving	1.839	261 μJ	13.611	88 μJ	147 μJ	21.667	47 μJ	79 μJ

Table 6.3: *Current drain of the different chipsets for their input range. Data sets for hashing and signing are 100 bytes*[2]*. Times are in ms*

common one in order to offer a good comparison of memory, communication and processing overhead on the different microprocessors.

6.3.3 Communication Processing

It is a matter of fact that when a node is member of a network every sent message is encrypted and therefore, AES-CCM encryption and decryption processing times –hence, consumption– should be considered.

6.4 Communication Overhead

The communication overhead of the proposed solution is definitely larger than other solutions which employ key pre-distribution or polynomial approaches in spite of a great security improvement thanks to ECDH. In Table 6.5, message sizes are summarized for the exchange messages described in Chapter 5. The sizes are displayed in four categories: with authentication information and no headers, with headers, without authentication information neither headers and with headers. Note that the headers of IEEE802.15.4 and UDP over 6loWPAN are 20 and 28 bytes for broadcast messages and unicast messages turning into 24 and 32 bytes after AES-CCM encryption due to the MIC. This frames were depictured in Figure 3.14 in comparison with the case without compression.

6.4.1 Single node overhead

The Equations (6.5), (6.6) and (6.7) describe the communication overhead per node since it joins the network, both in transmission and in reception. The length of the packets used

		New node	Cluster Head
Cortex M3	time	631.790 ms	636.469 ms
	avg	89.651 mJ	90.315 mJ
MSP430	time	2.016 s	2.031 s
	min	12.970 mJ	13.063 mJ
	max	21.777 mJ	21.933 mJ
CC430	time	4.917 s	4.908 s
	min	10.601 mJ	10.581 mJ
	max	17.879 mJ	17.844 mJ

Table 6.4: *Processing energy consumption for the different devices under different roles*

	Authentication		**Regular**	
Message	Payload	Headers	Payload	Headers
Pairing Request (P_{rq})	73	+20	73	+24
Pairing Response (P_{rs})	78	+28	77	+32
Pairing Proof (P_{pr})	6	+28	6	+32
Group Advertisement (G_{ad})	-	-	2	+24
Group Request (G_{rq})	6	+32	2	+32
Group Response (G_{rs})	25	+32	21	+32
Authentication Request (A_{rq})	-	-	30	+32
Authentication Response (A_{rs})	-	-	20	+32
Authenticity Request (T_{rq})	-	-	10	+24
Authenticity Confirm (T_{rs})	-	-	2	+32
Node Revocation (R_{rk})	-	-	26	+32

Table 6.5: *Message sizes (in bytes) for key exchange with and without authentication fields*

is reflected in Table 6.5. The other variables are: $\overline{R} = \overline{R_{col} + R_{slot}}$ the average number of retransmissions, N the number of neighbours, M the number of nodes that joined the network afterwards, N_n the number of new neighbouring nodes, η_{mid} the probability that the target node is between the CH and the GW, η_{sol} the probability a node requests a new key for a neighbour, η_{no} that a neighbour did not receive the group key and η_{known} the probability a pairing node is already paired with the target node.

$$O_n = O_{tx} + O_{rx} \tag{6.5}$$

$$
\begin{aligned}
O_{tx} = & \underbrace{(P_{rq} + P_{pr} + G_{rq})\overline{R}}_{\text{Pairing process}} + \underbrace{\lceil \frac{T_{\text{run}}}{T_{\text{pair}}} \rceil N \left((P_{rq} + P_{pr})(1 - \eta_{sol}) + P_{rs}\eta_{sol} \right) \overline{R}}_{\text{Link key renewal}} \\
& + \underbrace{(M + \lceil \frac{T_{\text{run}}}{T_{\text{group}}} \rceil) \cdot (G_{ad} + \eta_{\text{no}} G_{rq} + \eta_{\text{no}} N G_{rs}) \overline{R}}_{\text{Group key renewal}} \\
& + \underbrace{N_n \left(\eta_{\text{sol}}(P_{rs} + T_{rq}) + (1 - \eta_{\text{sol}})(P_{rq} + T_{rq} + P_{pr}) + N\eta_{\text{known}} T_{rs} \right) \overline{R}}_{\text{New neighbours in automated pairing}} \\
& + \underbrace{M\eta_{\text{mid}} N (A_{rq} + A_{rs})\overline{R} + \frac{N_n}{N}(1 - \eta_{\text{known}})(T_{rq} + T_{rs})\overline{R}}_{\text{Forward authentication and authenticity messages}}
\end{aligned}
\tag{6.6}
$$

$$
\begin{aligned}
O_{rx} = & \underbrace{(P_{rs} + G_{rs})\overline{R}}_{\text{Pairing process}} + \underbrace{\lceil \frac{T_{\text{run}}}{T_{\text{pair}}} \rceil N \left((P_{rq} + P_{pr})(\eta_{sol}) + P_{rs}(1 - \eta_{sol}) \right) \overline{R}}_{\text{Link key renewal}} \\
& + \underbrace{(M + \lceil \frac{T_{\text{run}}}{T_{\text{group}}} \rceil) \cdot (\eta_{\text{no}}(G_{ad} + G_{rs} + G_{rq}))\overline{R}}_{\text{Group key renewal}} \\
& + \underbrace{N_n \left(\eta_{\text{sol}}(P_{rq} + P_{pr}) + (1 - \eta_{\text{sol}})P_{rs} + (1 - \eta_{\text{known}})NT_{rq} \right) \overline{R}}_{\text{New neighbours in automated pairing}} \\
& + \underbrace{M\eta_{\text{mid}} N (A_{rq} + A_{rs})\overline{R} + \frac{N_n}{N}(1 - \eta_{\text{known}})(T_{rq} + T_{rs})\overline{R}}_{\text{Forward authentication and authenticity messages}}
\end{aligned}
\tag{6.7}
$$

Additionally, some time variables were included for exemplification, standing T_{pair} for the time to renew a key pair, T_{group} for the group key renewal time and T_{run} for the running time; in case they are decided to be renewed in a time fashion. As mentioned in Section Key Renewal (5.5.1), key renewal in a time basis is contemplated by the proposed solution but discouraged. Hence, for the rest of the document $T_{\text{pair}}, T_{\text{group}} \to \infty$ is assumed.

In Figure 6.8 and 6.9, the overhead of communication (as expressed in Equation (6.8) and (6.9)) in function of the total node joinings in the network is plotted for a network with an average number of 2 (blue), 5 (green) and 20 (red) neighbouring nodes and 20% of new neighbouring nodes (N_n). The probability of not having received a group key (η_{no}) is 75%. This probability is inversely related with the probability of having a known pair (η_{known}) which is considered 37.5%. The probability of a neighbour requesting key (η_{sol}) is 50% while the probability of being between the CH and GW (η_{mid}) is 0% in most of the networks. Finally the $\overline{R}$ is 1. The probabilities are quite realistic regarding Tado's network but they may be

adjusted for other network characterizations.

$$
\begin{aligned}
O_{tx} = 165 \\
&+ M \cdot (26 + \eta_{\text{no}} 34 + \eta_{\text{no}} N 53) \\
&+ N_n \left(\eta_{\text{sol}} 143 + (1 - \eta_{\text{sol}}) 169 + N \eta_{\text{known}} 34\right) \\
&+ M \eta_{\text{mid}} N (114) + \frac{N_n}{N} (1 - \eta_{\text{known}}) 68
\end{aligned}
\tag{6.8}
$$

$$
\begin{aligned}
O_{rx} = 162 \\
&+ M \eta_{\text{no}} \cdot 113 \\
&+ N_n \left(\eta_{\text{sol}} 135 + (1 - \eta_{\text{sol}}) 109 + N (1 - \eta_{\text{known}}) 34\right) \\
&+ M \eta_{\text{mid}} N (114) + \frac{N_n}{N} (1 - \eta_{\text{known}}) 68
\end{aligned}
\tag{6.9}
$$

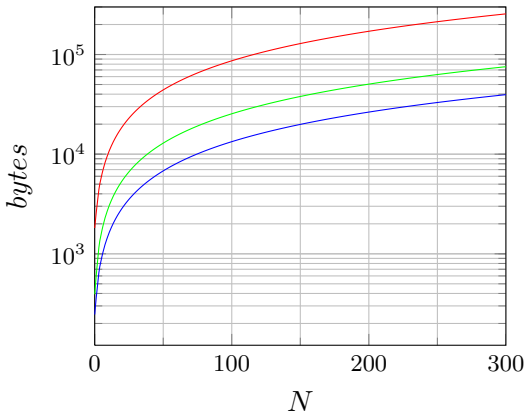


Figure 6.8: *Transmission communication overhead versus number of joining nodes for 2 neighbour networks (blue), 5 (green) and 20 (red)*

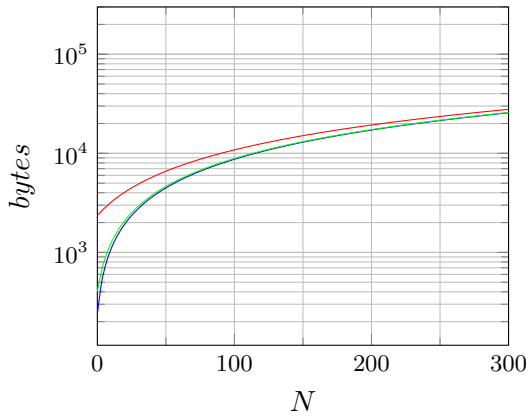


Figure 6.9: *Reception communication overhead versus number of joining nodes for 2 neighbour networks (blue), 5 (green) and 20 (red)*

In Figure 6.8, the communication overhead is drawn for transmission and in 6.9 the same information for reception is presented. A lineal growth in the overhead can be observed for both transmission and reception.

The very last node joining a 2 neighbours network, transmits an average of 246.1 bytes. When the network has an average of 5 neighbours per node, this figure grows up to 393.25 bytes for no new joins and to 1644.5 bytes when 5 new node join. These numbers demonstrate a very small footprint for small sized networks such the one in Tado in comparison to the regular network communication operations.

In comparison to polynomial approaches such as in [53], which has total communication overhead of 75 bytes independently of the number of joining nodes and the size of network, the given results may seem alarming when targeting larger networks. However, a closer look to some of the literature demonstrates that only direct communication to the CH has been considered. Additionally, those simulations do not contemplate neither key renewal, packet broadcasting nor the possibility of a neighbour node joining. Although this fact complicates the comparison between proposals, the given solution should not be compared to

pre-distributed keys or polynomial approaches as it offers a much higher level of security.

Nevertheless, looking at the regular communication overhead of the described protocol for regular communication, we can affirm that it is very low: only 4 bytes per any packet because of the AES-CCM-128 MIC versus the 6 bytes for a 30 bytes packet of SPINS[46].

6.4.2 Communication Energy Consumption

In Table 6.6, the energy consumption per node of the CC1101 and and CC430 chipsets is summarized. Several network configurations are used: 2, 5 and 20 neighbours per node networks and 0, 2 and 5 node joinings. The power values used are the ones in Table 6.3 considering 0dBm for transmission and maximum sensitivity for reception.

Neighbours per node	**Joins**	**Tx bytes**	**Rx bytes**	**CC1101**		**CC430**		
				min	max	min	max	battery
2 Neighbours	0	246,100	236,300	2,307	4,614	3,040	4,974	3,316
	2	508,100	405,800	4,375	8,750	5,762	9,429	6,286
	5	901,100	660,050	7,477	14,955	9,845	16,110	10,740
5 Neighbours	0	393,250	398,750	3,786	7,572	4,990	8,166	5,444
	2	893,750	568,250	7,008	14,017	9,223	15,093	10,062
	5	1644,500	822,500	11,842	23,683	15,573	25,483	16,989
20 Neighbours	0	1817,500	2358,500	19,933	39,867	26,293	43,025	28,683
	2	3510,500	2528,000	28,925	57,851	38,082	62,316	41,544
	5	6050,000	2782,250	42,413	84,827	55,765	91,252	60,835

Table 6.6: *Energy consumption by the RF chips and different network configurations. Message sizes in bytes and power consumption in mJ*

The Tado temperature sensor sends a message every minute if the temperature delta regarding the previous transmission is significant, or a message every 10 minutes when the delta is not significant. The message is about 16bytes of payload, thus a total of 38bytes including headers. In the worst case the node sends 19500KB of data per year and 1950KB in the best case. The rest of the messages are blocked, thus it cannot perform neighbour discovery nor process router advertisements. Therefore, the communication overhead considering a 5 neighbouring node network with 5 new node joins is about 0.08% of the annual communication for the least transmitting case. This statistic improves for the rest of nodes which exchange more messages. For instance, the CH communicates with the TAS once a minute using a 256bytes communication. This gross is 4.21MB of data per year and a 0.037% of the communication.

Communication Processing Energy

The processing overhead due to communication also has to be mentioned, especially in activities such as encryption, challenge solving, decryption, key computation and signatures. Given the Equations (6.6) and (6.6) and the Table 6.3 it is easy to obtain Equations (6.10) and (6.11). These are plotted in Figure 6.10 and Figure 6.11 for the same cases as in the

previous section.

$$\begin{aligned} O_{tx} = & \ 11.823 \\ & + M \cdot (0.004 + \eta_{\text{no}} 0.005 + \eta_{\text{no}} N 0.007) \\ & + N_n \left(\eta_{\text{sol}} 11.786 + (1 - \eta_{\text{sol}}) 11.771 + N \eta_{\text{known}} 0.005\right) \\ & + M \eta_{\text{mid}} N (0.016) + \frac{N_n}{N} (1 - \eta_{\text{known}}) 0.010 \end{aligned} \tag{6.10}$$

$$\begin{aligned} O_{rx} = & \ 11.842 \\ & + M \eta_{\text{no}} \cdot 0.016 \\ & + N_n \left(\eta_{\text{sol}} 11.766 + (1 - \eta_{\text{sol}}) 11.781 + N (1 - \eta_{\text{known}}) 0.005\right) \\ & + M \eta_{\text{mid}} N (0.016) + \frac{N_n}{N} (1 - \eta_{\text{known}}) 0.010 \end{aligned} \tag{6.11}$$

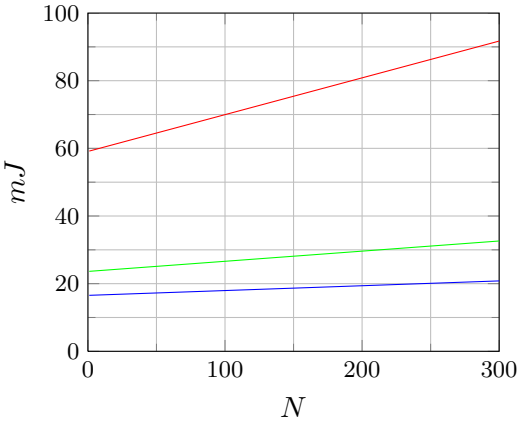


Figure 6.10: *Processing overhead for outgoing communication (including key generation) versus number of joining nodes for 2 neighbour networks (blue), 5 (green) and 20 (red)*

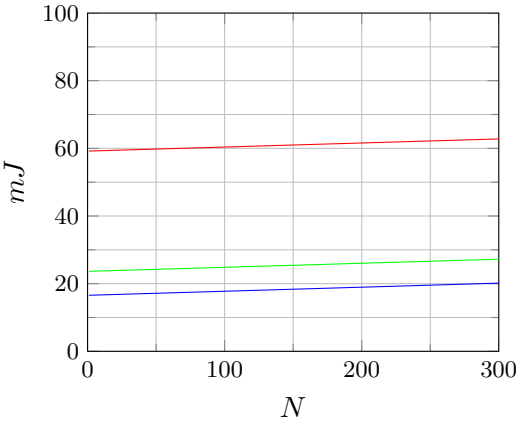


Figure 6.11: *Processing overhead for incoming communication (including key generation) versus number of joining nodes for 2 neighbour networks (blue), 5 (green) and 20 (red)*

A major influence of the processing overhead rather than the communication overhead can be observed, mostly due to the key computation procedures.

A single pairing on a MSP430 takes about 1.968s for the newcomer and 1.954s for the CH, which compared to the figures in [40] for the TelosB mote (assuming it is configured at 20MHz) show better performance than the TLS pairing (4.35s) but slightly worse than the TLS using IBC and ECDH (2.78s). However, in the mentioned reference no data is provided about the configuration of the microprocessor. Additionally, no memory figures are given to compare the memory overhead or processor configurations for a fine comparison.

6.5 Energy Consumption

Within this section, combined graphs of processing and communication energy consumptions are given for a better overview concerning the previous assumptions. Later on, a simulation

on the battery lifetime powered sensor is described.

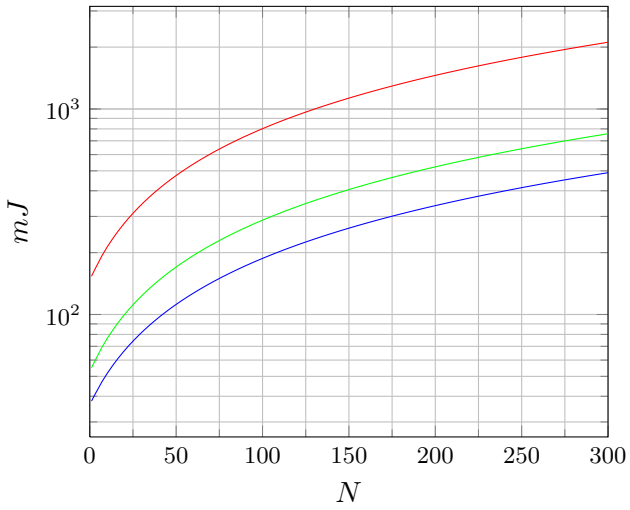


Figure 6.12: *Common consumption overhead versus number of joining nodes for 2 neighbour networks (blue), 5 (green) and 20 (red)*

In Figure 6.12, the addition of the processing and communication energy consumption for both outgoing and incoming packets is plotted. It can be seen that the energy consumption is mainly driven by processing overhead due to the key computation. More efficient algorithms for key computation (elliptic curve point multiplication) shall reduce the overall consumption.

6.5.1 Battery Life of the CC430

It is important to see the impact of the proposed solution on the design battery life of the Tado sensor (CC430 platform). The former is powered by two AAA batteries and a solar cell, where the solar cell is a secondary power supply and the batteries are the main power source. The rechargeable NiMH batteries are recommended.

According to the Energizer HR03 datasheet[23] on typical discharge characteristics, the energy available in the battery is the integral of the curve at 80mA discharge rate. The interesting range is between 1.3 and 1.1V as the CC430 input range varies between 3.6V and 2.2V and the device is powered by two serial batteries. Approximating the curve given in the datasheet by a line between these two values, an energy of about 3060J (W_b) is obtained. Neither information regarding self-discharge effects of this battery when connected to line nor a good approximation for the general case have been found, suggesting no self-discharge or marginal discharge effects when connected. Therefore, no self-discharge is considered in this simulation.

The CC430 has a standby consumption of about 2μA[24] when in Low Power Mode 3 (LPM3), this permits the lowest consumption possible while holding the RAM data and allowing wake-ups due to interrupts. The platform will keep the processor in LPM3 and waking up once per minute to compute and broadcast the temperature. If this temperature has a relative variance of 0.3^{o}C regarding the last transmitted temperature, the node sends the temperature to the CH, otherwise it goes to LPM3 again. The temperature is transmitted mandatorily once every 10 minutes.

The report data consists in 16 bytes of UDP payload, which includes error flags and status messages. This packet is encapsulated by IEEE802.15.4 and 6loWPAN headers, which add up

to 24 bytes, making a 40 bytes transmission. Assuming that the sensor is only paired once, that it needs 5ms to compute the temperature readings, a mean supply voltage of 2.4V and the current consumptions are stated in 6.3. The ideal platform consumption is as stated in Equation (6.12).

$$\begin{aligned} W_m &= compute + encrypt + transmit + sleep \\ &= 12.120\mu J + 5.613\mu J + 276.480\mu J + 287.328\mu J = 581.541\mu J \end{aligned} \tag{6.12}$$

The sleep mode energy is computed taking into account the temperature reading time, the encryption time and the transmission and subtracting it from one minute: $t_{\text{sleep}} = 60s - 5ms - 2.32ms - 6.4ms = 59.86s$ and therefore $W_{sleep} = VI_{LPM3}t_{sleep} = 287.328\mu J$. Thus, as it can be seen, is the main contribution to the energy consumption of the platform regardless of being in "standby" mode.

Given the consumption values in Table 6.6 and in Table 6.4, for the simulations presented in Section Communication Overhead the the lifetime of the batteries (Equation (6.13)) can be plotted as in Figure 6.13. In (6.13), W_p stands for the energy consumption of the described key generation and distribution protocol.

$$T = \frac{W_b - W_p}{W_m} \approx 10\ years \tag{6.13}$$

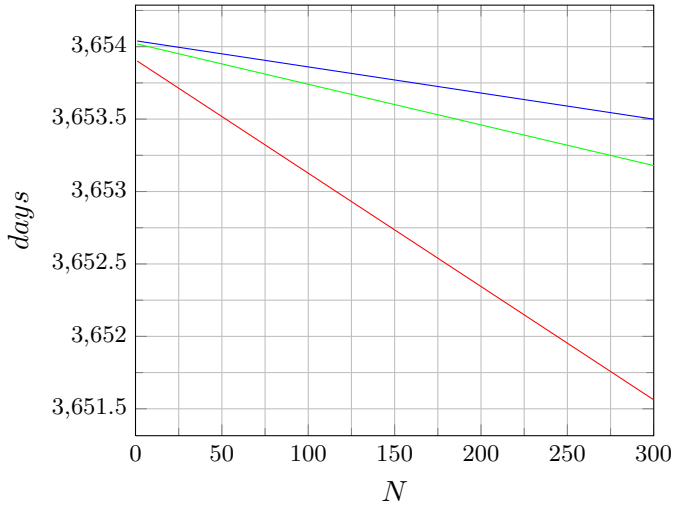


Figure 6.13: *Battery life of the CC430 using 2 AAA batteries on a 2 neighbours network (blue), 5 (green) and 20 (red)*

As it can be seen in the previous graphic, the energy consumption due to communication does influence the life of the sensor device in approximately 2-3 days in 10 years for the most consuming configuration. This demonstrates that the communication overhead of the protocol is minimal and it is a good approach for solving the key distribution problem.

Please note that in this estimation only pure processor consumptions have been considered. The sensing board also has a temperature sensor, operational amplifiers and other circuitry which may reduce notably the battery life dramatically. Additionally, as already mentioned, it is not guaranteed that the batteries do not present a self discharge factor even if not specified by the manufacturer. Furthermore, neither the RDC effects, such as burst retransmissions; retransmissions due to collision or slot; nor other physical channel effects have been considered.

Albeit these effects may scale considerably the transmission energy consumption, the main contributor is the key computation and therefore the difference may not be significant. The design of this platform claims a battery life of about 2 years without solar cell assistance considering all the electronic components and a transmission every minute.

This simulation evidences that the proposed solution only influences the overall energy consumption slenderly and it will not reduce the battery life significantly, which makes it attractive for all the sensors within Tado network. Hence, the obtained result is compatible with the sensor platform design.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this master thesis, a solution to the key generation and distribution problem has been proposed and developed, which uses Elliptic Curve Diffie-Hellman and Key Derivation Function 2 for key generation and symmetric encryption to ease the computation requirements of Elliptic Curve Cryptography. All the security goals described in the Objectives section have been achieved: *confidentiality*, *integrity*, *strong authentication*, *non repudiation*, *authorization* and *data freshness*. The resilience of the proposed scheme has been proved against the most adversaries: eavesdropping, fraudulent devices, message replies and man-in-the-middle. It has been argued that security against physical attacks is difficult to achieve in this kind of networks as the deletion of secret information upon deployment may lead to unusable devices, which is not attractive for consumer products or mobile nodes.

A complete study of the current standards and technologies available for WSN demonstrated that the key exchange over multicast UDP presented the lowest overhead due to the lower protocols; and that AES-CCM offers integrity, authenticity and confidentiality of the communication with a minimum additional overhead of 32 bits per transmitted frame which offers a good level of resilience against brute force attacks. Furthermore, it have been seen that a key length of 128 bits for AES offers a good level of security.

The huge impact on the memory footprint due to compiler optimizations has been documented, which is not possible to control, differentiating in about 3KB the Relic-Toolkit size between the MSP430 and the Cortex M3 microprocessors. On the other hand, it was proved that intelligent code optimization can lead to important reductions of memory footprint and processor time. This optimizations were: the usage of the microprocessor peripherals as much as possible and to reduce the code footprint, since it is the bottleneck in many WSN applications.

The work done reveals that Elliptic Curve Cryptography is affordable by using hybrid schemes with a relatively small processing and communication overhead. More labor must be executed on the cryptographic libraries as it still represents the 13.57% of the total flash memory available. It can also be noticed that the size of the SHA1 hashing algorithm is the 5.57% of the flash of the CC430. Hence, more efficient lightweight hashing functions should be researched.

The developed key generation hybrid schema brings together the strength of ECDH with the possibility of using symmetric encryption and non-abusive processor techniques for

random number generation. Furthermore, the use of partial solution of challenges and a Trusted Authentication Server permit strong authentication and detects Man-in-the-Middle attacks while keeping the communication overhead at a minimum. Finally, it uses symmetric AES-CCM-128 encryption as recommended by the IEEE802.15.4 standard. Nevertheless, an Intrusion Detection System is required for the detection of physical attacks an for network health reports.

Key generation procedures have demonstrated a big processing impact due to point operations monopolising the microprocessor during 4.845s. This can be a limiting factor for low featured processors nodes within networks with a high variance of members. A processing consumption of 153mJ was acquired after a simulation of a network for 20 neighbouring nodes per node with 300 posterior joins. For consumer small networks, such as Tado's, it has revealed an energy consumption as low as 40-55mJ for the CC430.

The security achievements of this solution make it suitable for any size of network even with limited nodes. The developed protocol is flexible enough to include custom additions, which allows its use in not-hierarchical networks by allowing every node to generate a group key, while still ensuring strong authentication. The experimentation has revealed that the processing times for new link key generation are affordable since typically the networks do not grow indefinitely and have a reduced number of neighbouring nodes. The lack of link key renewal makes the key generation operation a one-time operation attractive for WSN.

7.2 Future Work

Improvements in memory footprint and processing time by the use of hardware peripherals have been documented[35, 47]. These require about three orders of magnitude less energy than the software implementation: about $12mJ$ for the software version in contrast to $11\mu J$ for the fastest of the accelerators reviewed. Hence, the use of hardware accelerators for ECC operations is encouraged. This reduces the main contribution to the power drain in the proposed solution and makes it, therefore suitable for more constrained nodes. Although the code optimizations –as proposed in [6]– also prove a reduction of the computation time on the MSP430 boards, these do not reduce the code size as much as a hardware accelerator. This effect has has been demonstrated by using the AES peripheral on the CC430. Moreover, these low level code optimizations are processor specific makes the portability to different platforms not an easy task.

Additionally, it must be mentioned that this work presents only simulations and figures for chipset exclusive consumption. More accurate calculations shall be done, considering the peripherals consumption and other circuitry of the boards as well as using more accurate measurement equipment. Despite of this, a bias and additional currents drained by the boards can be easily added to the presented results for an approximated estimation.

The physical attack analysis (Section 6.1) revealed important weaknesses in the schema, making an IDS presence mandatory. Other approaches shall be researched in order to grant commercial products of a good physical resilience against physical attacks. Another reported weakness is the cooperative authenticity report feature. Despite the fact that it lowers the communication overhead of the CH, it may lead to fraudulent node access to the network if two or more neighbouring nodes are compromised. Hence, further research in this direction is advocated as well.

Bibliography

[1] "Contiki: The open source operating system for the internet of things," http://www.contiki-os.org/. [Online]. Available: http://www.contiki-os.org/

[2] "IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)," *IEEE Std 802.15.4-2003*, pp. 0_1 –670, 2003.

[3] "IEEE standard for local and metropolitan area networks–Part 15.4: Low-rate wireless personal area networks (LR-WPANs) amendment 3: Physical layer (PHY) specifications for low-data-rate, wireless, smart metering utility networks," *IEEE Std 802.15.4g-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1 – 252, 2012.

[4] D. F. Aranha and C. P. L. Gouvêa, "RELIC is an Efficient LIbrary for Cryptography," http://code.google.com/p/relic-toolkit/.

[5] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia, "QUARK: a lightweight hash," in *Proceedings of the 12th international conference on Cryptographic hardware and embedded systems*, ser. CHES'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 1 – 15.

[6] J. Ayuso, L. Marin, A. Jara, and A.F.G. Skarmeta, "Optimization of public key cryptography (rsa and ecc) for 16-bits devices based on 6lowpan," *1st International Workshop on the Security of the Internet of Things, Tokyo, Japan*, 2010.

[7] Kenneth Barr and Krste Asanović, "Energy aware lossless data compression," in *Proceedings of the 1st international conference on Mobile systems, applications and services*, ser. MobiSys '03. New York, NY, USA: ACM, 2003, pp. 231 – 244.

[8] W. Bechkit, "New key management schemes for resource constrained wireless sensor networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, Jun. 2011, pp. 1 – 3.

[9] R. Blom, "An optimal class of symmetric key generation systems," in *Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 335 – 338.

[10] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger, "Biclique cryptanalysis of the full AES," in *Proceedings of the 17th international conference on The*

Theory and Application of Cryptology and Information Security, ser. ASIACRYPT'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 344 – 371.

[11] Haowen Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *2003 Symposium on Security and Privacy, 2003. Proceedings*, May 2003, pp. 197 – 213.

[12] Chih-Chun Chang, S. Arafa, and S. Muftic, "Key establishment protocol for wireless sensor networks," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, Oct. 2007, pp. 1 – 6.

[13] R. Di Pietro, L.V. Mancini, and A. Mei, "Efficient and resilient key discovery based on pseudo-random key pre-deployment," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, Apr. 2004, p. 217.

[14] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei, "Random key-assignment for secure wireless sensor networks," in *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, ser. SASN '03. New York, NY, USA: ACM, 2003, pp. 62 – 71.

[15] Laurent Eschenauer and Virgil D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 41 – 47.

[16] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), "ISO/IEC 18033-2:2006, information technology – security techniques – encryption algorithms – part 2: Asymmetric ciphers," 2006.

[17] S. D. Galbraith and N. P. Smart, "Evaluation report for CRYPTREC: Security level of cryptography - ECDLP mathematical problem," Tech. Rep., 2008.

[18] O. Garcia-Morchon, T. Falck, T. Heer, and K. Wehrle, "Security for pervasive medical sensor networks," in *Mobile and Ubiquitous Systems: Networking Services, MobiQuitous, 2009. MobiQuitous '09. 6th Annual International*, Jul. 2009, pp. 1 – 10.

[19] Jian Guo, Thomas Peyrin, and Axel Poschmann, "The PHOTON family of lightweight hash functions," vol. 6841. LNCS, 2011, pp. 222 – 239.

[20] Andrei Gurtov, *Host Identity Protocol (HIP): towards the secure mobile Internet.* Wiley, 2008.

[21] Qiang Huang, Johnas Cukier, Hisashi Kobayashi, Bede Liu, and Jinyun Zhang, "Fast authenticated key establishment protocols for self-organizing sensor networks," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, ser. WSNA '03. New York, NY, USA: ACM, 2003, pp. 141 – 150.

[22] Jonathan Hui and Pascal Thubert, "Compression format for IPv6 datagrams over IEEE 802.15.4-based networks," http://tools.ietf.org/html/rfc6282.

[23] Energizer Holdings Inc., "Energizer HR03-800 nimh rechargeable battery datasheet," Jan. 2013. [Online]. Available: http://data.energizer.com/PDFs/HR03-800_EU.pdf

[24] Texas Instruments Inc., "Texas instruments CC430f5137 datasheet," Dec. 2011. [Online]. Available: http://www.ti.com/lit/gpn/cc430f5137

[25] Rongrong Jiang and Tieming Chen, "A bilinear pairing-based key management scheme for wireless sensor networks," in *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, Dec. 2011, pp. 670 – 673.

[26] A. Khurri, D. Kuptsov, and A. Gurtov, "On application of host identity protocol in wireless sensor networks," in *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, Nov. 2010, pp. 358 – 345.

[27] Neal Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, p. 203, Jan. 1987.

[28] ——, "CM-Curves with good cryptographic properties," in *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '91. London, UK, UK: Springer-Verlag, 1992, pp. 279 – 287.

[29] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun, "Caveat eptor: A comparative study of secure device pairing methods," in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, Mar. 2009, pp. 1 – 10.

[30] Nandakishore Kushalnagar, Gabriel Montenegro, David E. Culler, and Jonathan W. Hui, "Transmission of IPv6 packets over IEEE 802.15.4 networks," http://tools.ietf.org/html/rfc4944.

[31] Paul J. Leach, Tim Berners-Lee, Jeffrey C. Mogul, Larry Masinter, Roy T. Fielding, and James Gettys, "Hypertext transfer protocol – HTTP/1.1," http://tools.ietf.org/html/rfc2616.

[32] Yue Li and T. Newe, "Key exchange protocol for wireless sensor network: Formal verification using CSN modal logic," in *Sensors Applications Symposium, 2008. SAS 2008. IEEE*, Feb. 2008, pp. 193 – 198.

[33] J. Lopez and R. Dahab, *Fast multiplication on elliptic curves over GF(2m) without precomputation*, 1999.

[34] Yasir Arfat Malkani, Dan Chalmers, and Ian Wakeman, "A framework for secure device pairing by demonstration of physical proximity," 2010.

[35] K. McCusker and N.E. O'Connor, "Low-energy symmetric key distribution in wireless sensor networks," *Dependable and Secure Computing, IEEE Transactions on*, vol. 8, no. 3, pp. 363 – 376, Jun. 2011.

[36] Zhang Min-Qing, Fu Wen-Hua, and Li De-Long, "A new key management scheme based on secret information for WSN," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, May 2011, pp. 518 – 521.

[37] H. Modares, R. Salleh, and A. Moravejosharieh, "Overview of security issues in wireless sensor networks," in *Computational Intelligence, Modelling and Simulation (CIMSiM), 2011 Third International Conference on*, Sep. 2011, pp. 308 – 311.

[38] R. Moskowitz and P. Nikander, "Host identity protocol (HIP) architecture," http://tools.ietf.org/html/rfc4423, May 2006.

[39] R. Moskowitz, P. Nikander, Ed. P. Jokela, and Ed. P. Jokela, "Host identity protocol," http://tools.ietf.org/html/rfc5201, Apr. 2008.

[40] R. Mzid, M. Boujelben, H. Youssef, and M. Abid, "Adapting TLS handshake protocol for heterogenous IP-based WSN using identity based cryptography," in *Communication in Wireless Environments and Ubiquitous Systems: New Challenges (ICWUS), 2010 International Conference on*, Oct. 2010, pp. 1 – 8.

[41] N. Nehra and R.B. Patel, "MASLKE: mobile agent based secure location aware key establishment in sensor networks," in *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, Dec. 2008, pp. 1 – 6.

[42] S. Nitinawarat, Chunxuan Ye, A. Barg, P. Narayan, and A. Reznik, "Secret key generation for a pairwise independent network model," *Information Theory, IEEE Transactions on*, vol. 56, no. 12, pp. 6482 – 6489, Dec. 2010.

[43] A.N.M. Noman, "A generic framework for defining security environments of wireless sensor networks," in *Electrical and Computer Engineering, 2008. ICECE 2008. International Conference on*, Dec. 2008, pp. 924 – 929.

[44] National Institute of Standards and Technology, "Recommended elliptic curves for government use," Jul. 1999. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf

[45] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song, "The TESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, no. Summer, 2002.

[46] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar, "SPINS: security protocols for sensor networks," in *Seventh Annual International Conference on Mobile Computing and Networks (MobiCOM 2001)*, Rome, Italy, Jul. 2001.

[47] Steffen Peter, Peter Langendorfer, and Krzysztof Piotrowski, "Public key cryptography empowered smart dust is affordable," *Int. J. Sen. Netw.*, vol. 4, no. 1/2, pp. 130 – 143, Jul. 2008.

[48] Pyrgelis Apostolos, "Cryptography and security in wireless sensor networks," Department of Computer Engineering and Informatics University of Patras, Greece, Oct. 2009.

[49] Qamar Toheed and Hassan Razi, "Asymmetric-key cryptography for contiki," Ph.D. dissertation, Chalmers University of Technology, University of Gothenburg, Department of Computer Science and Engineering, Sweden, Jul. 2010.

[50] M Rahman and K El-Khatib, "Secure time synchronization for wireless sensor networks based on bilinear pairing functions," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2010.

[51] M.A. Rahman and M.K. Debnath, "An energy-efficient data security system for wireless sensor network," in *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conference on*, Dec. 2008, pp. 381 – 386.

[52] Inc. RIGOL Technologies, "DS1000D/E series oscilloscope specifications," 2008.

[53] An-Ni Shen and Song Guo, "Key re-establishment protocols in hierarchical wireless sensor networks," in *Wireless Conference, 2009. EW 2009. European*, May 2009, pp. 184 – 188.

[54] An-Ni Shen, Song Guo, and Hung-Yu Chien, "An efficient and scalable key distribution mechanism for hierarchical wireless sensor networks," in *Sarnoff Symposium, 2009. SARNOFF '09. IEEE*, Apr. 2009, pp. 1 – 5.

[55] K.G. Srinivasa, V. Poornima, V. Archana, C. Reshma, K.R. Venugopal, and L.M. Patnaik, "Combinatorial approach to key generation using multiple key spaces for wireless sensor networks," in *Advanced Computing and Communications, 2008. ADCOM 2008. 16th International Conference on*, Dec. 2008, pp. 279 – 284.

[56] Sushmita Ruj and Bimal Roy, "Key predistribution using combinatorial designs for grid-group deployment scheme in wireless sensor networks," 2009.

[57] A. Tufail, A. Ali, and Ki-Hyung Kim, "A reliable key management scheme for wireless sensor networks," in *Optical Internet (COIN), 2010 9th International Conference on*, Jul. 2010, pp. 1 – 3.

[58] Jean-Philippe Vasseur and Adam Dunkels, *Interconnecting Smart Objects with IP - The Next Internet*. Morgan Kaufmann, 2010.

[59] Jonathan Voris, Nitesh Saxena, and Tzipora Halevi, "Accelerometers and randomness: perfect together," in *Proceedings of the fourth ACM conference on Wireless network security*, ser. WiSec '11. New York, NY, USA: ACM, 2011, p. 115–126.

[60] Qian Wang, Hai Su, Kui Ren, and Kwangjo Kim, "Fast and scalable secret key generation exploiting channel phase randomness in wireless networks," in *INFOCOM, 2011 Proceedings IEEE*, Apr. 2011, pp. 1422 – 1430.

[61] Doug Whiting, Niels Ferguson, and Russell Housley, "Counter with CBC-MAC (CCM)," https://tools.ietf.org/html/rfc3610.

[62] Jaesung Yoo, Yunho Lee, and Dongho Won, "An improved key establishment scheme for wireless sensor network," in *Proceedings of the 6th International Conference on Future Internet Technologies*, ser. CFI '11. New York, NY, USA: ACM, 2011, pp. 70 – 71.

[63] Nanrun Zhou, Qiongxi Jiang, and Xun Chen, "Identity-based key management scheme with provable security for wireless sensor networks," *Journal of Information & Computational Science*, vol. 8, no. 14, pp. 3075 – 3081, 2011.

[64] Tanveer A. Zia and Albert Y. Zomaya, "A lightweight security framework for wireless sensor networks," *Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications*, vol. 2, pp. 53 – 73, Sep. 2011.