

INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME

Project n°: FP6-IST-033563



SMEPP

Secure Middleware for Embedded Peer-to-Peer Systems

WP1

D1.2 Security Requirements of EP2P

Applications

Author(s): TUG, I2R, SIEM, UMA
Status -Version: 1.03
Date: July 4, 2007
Distribution - Confidentiality: Public
Code: D1.2_v1.03.doc

Disclaimer

This document contains material, which is the copyright of certain SMEPP contractors, and may not be reproduced or copied without permission. All SMEPP consortium partners have agreed to the full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

The SMEPP Consortium consists of the following companies:

Participant no.	Participant name	Participant short name	Country
1 (Coordinator)	Universidad de Málaga	UMA	Spain
2	Tecnatom, S.A.	TEC	Spain
3	Technische Universität Graz	TUG	Austria
4	Siemens AG	SIEM	Germany
5	Valtion Teknillinen Tutkimuskeskus	VTT	Finland
6	Università di Pisa	UPI	Italy
7	Telefónica I+D	TID	Spain
8	Institute for Infocomm Research	I2R	Singapore

The information in this document is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Document Revision History

Date	Issue	Author/Editor/Contributor	Summary of main changes
March 8, 2007	0.1	Manfred Aigner	Initial proposal for ToC
March 13, 2007	0.2	Stefan Tillich	Document skeleton for initial revision (M6)
March 23, 2007	0.3	Stefan Tillich	Goals for cryptographic protection in EP2P
April 04, 2007	0.4	Jianying Zhou	Threats for EP2P Systems
April 17, 2007	0.5	Anton Kargl, Rodrigo Roman	Authentication & Non-repudiation; Crypto and Threats in EP2P
April 23, 2007	0.6	Nuria Sanz Martin, Erwin Heß, Udo Payer, Stefan Kraxberger, Stefan Tillich	Integrate comments of TID; New version of Authentication & Non-repudiation; Protection of EP2P systems; Side-channel attacks; Update of overview
April 24, 2007	0.7	Stefan Kraxberger, Stefan Tillich	Update of section “Protection of EP2P systems”; Minor editorial changes
April 27, 2007	0.8	Stefan Tillich	Formatting, correct typos
May 11, 2007	0.9	Stefan Tillich	Restructuring following classification of EP2P system aspects
May 30, 2007	1.0	Jianying Zhou, Stefan Kraxberger, Udo Payer, Tan Han Chiang	Classification of EP2P systems, modification of P2P threats
July 4, 2007	1.03	Stefan Kraxberger, Udo Payer	Application level security adapted to D3.2

Table of contents

1	Introduction.....	6
1.1	Purpose.....	6
1.2	Audience.....	6
2	Overview	7
2.1	Purpose of the Project.....	7
2.2	Objectives and Success Criteria of the Project	7
3	General Application Security Requirements.....	8
3.1	Confidentiality	9
3.2	Data Integrity	11
3.3	Authentication.....	12
3.4	Non-Repudiation.....	14
3.5	References	16
4	Embedded P2P Systems and Applications.....	18
4.1	Aspects of Embedded P2P Systems	18
4.1.1	P2P Network Types	18
4.1.1.1	Centralized P2P network such as Napster.....	19
4.1.1.2	Decentralized P2P network such as KaZaA.....	20
4.1.1.3	Structured P2P network such as CAN	20
4.1.1.4	Unstructured P2P network such as Gnutella	21
4.1.1.5	Hybrid P2P network such as JXTA.....	21
4.1.2	Peer capabilities and environment constraints.....	21
4.1.2.1	Platform Resources.....	21
4.1.2.2	Communication Facilities	22
4.1.2.3	Device Mobility.....	23
4.1.2.4	Possible Combinations.....	24
4.2	Threats.....	25
4.2.1	P2P Network Threats	25
4.2.1.1	Threats due to Restricted Resources	25
4.2.1.2	Communication Facilities Threats	26
4.2.1.3	Device Mobility Threats	27
4.2.2	Other Threats	28
4.2.3	References	31
5	Protection of EP2P Systems	33
5.1	Data Origin Authentication.....	33
5.1.1	Authentication Mechanisms	34
5.1.1.1	Secret Information Based Asymmetry.....	34
5.1.1.2	Time Based Asymmetry.....	34
5.1.1.3	Hybrid Approaches.....	35
5.1.1.4	Signature Propagation.....	35
5.1.1.5	Signature Dispersal.....	36
5.1.1.6	Differed Signing	36
5.1.2	Requirements.....	36
5.2	Secure EP2P Networking	37
5.3	Communication Security	38
5.4	Application Level Protection	39
5.5	Security Management and Enforcement.....	41
5.5.1	Security- and Trust levels.....	41
5.5.2	Specification, Decision and Delegation	41

- 5.5.3 Deployment and Enforcement.....42
- 5.5.4 Auditing and Monitoring.....42
- 5.6 Protection Against Side-Channel Analysis.....43
- 5.7 References43
- 6 Using Cryptography to Avoid Threats in EP2P Systems 47**
- 6.1 Cryptography and Middleware/Application Threats in EP2P Systems.....47
- 6.2 Suitability of Cryptography in Constrained EP2P Devices49
- 6.3 References50
- 7 Glossary 52**

1 Introduction

1.1 Purpose

This document describes the security requirements for embedded peer-to-peer (EP2P) applications. It introduces the main threats against the security of EP2P systems and the principal security assurances that cryptographic measures can provide to thwart these threats. Then this document discusses different techniques to secure EP2P systems through cryptographic and non-cryptographic methods.

1.2 Audience

The audience for this document includes the project management, system analysts, system designers and testers.

2 Overview

2.1 Purpose of the Project

Embedded Peer-to-Peer Systems (EP2P) represent a new challenge in the development of software for distributed systems. These systems have brought about an important revolution in distributed computing paradigms, now that the roles of client and server, which are the basis of the most widely used distributed computation models, are disappearing. The new scenario consists of systems in which all the elements of the network are symmetrical and in most cases, the mechanisms of communication are not based on pre-existing infrastructures, but rather on dynamic ad-hoc networks among peers. At the same time, the recent technological advances in short distance wireless communications have opened up new areas of application which represent important technological challenge. In addition, these systems are extremely vulnerable against any type of internal or external attacks, due to resource constraints, lack of tamper-resistant packaging, and the nature of open and public communication channels.

One of the key factors in the success of these systems is the possibility of abstracting all these problems by means of convenient middleware. This middleware should hide the complexity of the underlying infrastructure while providing open interfaces to third parties for application development. The development of such a middleware is challenging, since besides the disappearance of the roles of client and server, other critical requirements appear, which have to be supported by these infrastructures (mobility, new security problems, discovery and localization protocols, new quality of software criteria, etc). The main objective of this project is to develop a new secure and generic middleware, based on a new network centric abstract model for EP2P systems. Its suitability will be demonstrated by the development of two real-life applications in the domains of Environmental Monitoring in Industrial Plants and Home Systems and Mobile Telephony.

2.2 Objectives and Success Criteria of the Project

The main objective of this project is to develop a new middleware, based on a new network centric abstract model, specially designed for the above described scenario, and trying to overcome the main problems of the currently existing domain specific middleware proposals. The middleware will be secure, generic and highly customizable, allowing for its adaptation to different devices (from PDAs and new generation mobile phones to embedded sensor actuator systems) and domains (from critical systems to consumer entertainment or communication). Its suitability will be demonstrated by the development of two different innovative real-life applications in the domains of Home Systems and Mobile Telephony and Environmental Monitoring in Industrial Plants.

SMEPP self-assessment will be implemented at two different levels. In a first level, the project results will be evaluated for each of the project objective areas with respect to the success criteria defined in this section. This evaluation will be carried out at the end of each iteration of the project, following an incremental approach. The second level of self-assessment will be carried out in the context of an independent validation task, which will assess the project success in a global manner. This task is included in a separate validation Work-Package (WP6) where we will evaluate all the middleware components and associated tools during the development of the applications and with respect to the concrete objectives and characteristics established in the requirement phase of the project.

3 General Application Security Requirements

In order to define security requirement for EP2P applications it is necessary to understand the principal security assurances which can be achieved with a systematical application of cryptographic methods.

The Handbook of Applied Cryptography [1] defines *cryptography* as “the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication”. The four commonly used categories of information security objectives are *confidentiality*, *data integrity*, *authentication*, and *non-repudiation*. Cryptographic algorithms (primitives) and protocols have been designed to provide security assurances for objectives in all four categories.

Cryptographic primitives can be subdivided into three broad categories, each fit to provide specific security aspects. These categories are unkeyed primitives (e.g. hash functions), symmetric-key primitives (e.g. block and stream ciphers) and public-key primitives (e.g. public-key ciphers). Cryptographic protocols (for e.g. digital signatures) use these primitives to provide security assurances for two or more entities.

The security challenges for traditional computer networks have been and continue to be manifold. “Grown” networks like the Internet are exposed to multitudes of security threats, which can often be only mitigated to a certain degree by added-on security solutions. Decades after its creation, the Internet still suffers from the neglect of security issues, which has taken place in the early phases of its development.

Embedded computing devices appear in a steadily growing number of everyday appliances, and are equipped with increasingly powerful network connection facilities. Networks of embedded devices will be able to provide exciting new applications; but they will also face the same security challenges as traditional computer networks. Embedded networks (e.g. sensor networks) will be deployed to protect valuable assets, or might present a valuable asset in themselves (e.g. “ambient intelligence”). Therefore they will be exposed to attacks, e.g. to circumvent their protection or gain unauthorized access to their services. Moreover, as embedded devices are limited in their resources, a proper defense against attacks will be difficult. Only a security-aware design approach based on strong cryptographic methods can provide a sound basis for a reasonable set of provided security assurances in embedded networks.

Embedded systems often use a wireless network infrastructure which is inherently based on a broadcast medium. Standards exist for Wireless Local Area Networks (WLAN), e.g. IEEE 802.11g [2] and Wireless Personal Area Networks (WPAN), e.g. IEEE 802.15.1 (Bluetooth) [3], IEEE 802.15.4 [4] (base for ZigBee [5]). Another set of standards for wireless communication stems from mobile telephony, e.g. UMTS [6]. Any broadcast medium is inherently vulnerable to eavesdropping, jamming and message injection, so there is a strong need for security solutions for wireless communication.

What security assurances have actually to be provided is highly dependent on the application at hand. For example, in some applications it might be sufficient to guarantee authenticity of sent messages, while in other applications it might also be necessary to protect the contents of the messages. For some applications it may even be desirable that senders of messages cannot later deny to have sent it. A secure middleware must therefore be able to provide solutions for all potential security threats. Moreover, as security comes at the price of increased processing workload, memory requirements, power consumption, as well as network traffic, and as embedded devices are normally limited in their resources, it is desirable to adapt the security measures to the security demands of the application.

Despite its deficiencies, the Internet can serve as a valuable source of experience for networking security. Security solutions can be found at different layers of the network protocol stack. In the Internet today, security at the data-link layer is rarely found. One of its main uses is for secure dial-up using the point-to-point protocol (PPP) [7]. The most important security protocols are situated at higher layers of the network protocol stack. IPsec [8] – as the name implies – is a family of protocols for securing the Internet's network layer protocol, the Internet protocol (IP). Each IP packet can be authenticated and optionally also encrypted. Above the transport layer, the Secure Sockets Layer (SSL) protocol [9] and its advancement, the Transport Layer Security (TLS) protocol [10] can provide security. At the application layer, there are solutions like Secure Shell (SSH) [11], Pretty Good Privacy (PGP) [12], and Secure / Multipurpose Internet Mail Extensions (S/MIME) [13], which are used to secure network services like remote login or e-mail.

Security protocols like IPsec and TLS do not mandate the use of a specific set of cryptographic primitives. Entities which wish to exchange data securely have to first establish a common set of cryptographic primitives (*cipher suite*), which is available at both sides. Negotiation of a cipher suite provides both scalability of security level (e.g. selection of key size) as well as flexibility in regard to used cryptographic primitives. This might be important in the event of the development of new cryptanalytic attacks against specific cryptographic primitives.

In embedded devices it is often infeasible to support different cipher suites, mainly due to restrictions in program memory size. Severely restricted devices like sensor nodes might even be unable to support more than a single, light-weight cryptographic algorithm. In environments where communication is costly, it would also be very impractical to incur communication overhead with a cipher suite agreement. For low-cost devices it might therefore be necessary to base all cryptographic protocols on a fixed – and often small – set of cryptographic primitives. Selection of these primitives must be very careful in regard to both the degree of security as well as the suitability for implementation in a restricted environment.

As the SMEPP project focuses on the peer-to-peer paradigm for embedded networks, we will highlight in this section the goals for cryptographic protection in such a setting. Starting from the situation in traditional systems, we will highlight similarities and differences for the EP2P domain for the four basic categories of security objectives.

3.1 Confidentiality

Confidentiality deals with keeping the contents of information restricted to a set of authorized entities. In computer networks, a common goal is to transfer data from one entity to another over a potentially insecure network infrastructure, without disclosing the data to third parties. If wireless networking is employed, eavesdropping is an inherent threat as it is virtually impossible to establish physical security of the medium (e.g. wireless networking within a secluded area of operation protected by a physically inaccessible perimeter).

As confidentiality is defined with respect to authorized entities, its realization is dependent on the reliable authentication of those entities which can access the protected data. Therefore, confidentiality should not be seen as a stand-alone service, but as an add-on service in a security framework.

Confidentiality is normally achieved by the use of symmetric-key ciphers, where sender and receiver possess the same (or trivially related) keys. The sender encrypts the message,

and sends it over the network. Only receivers with the same key can then decrypt the message. In this case, the possession of the symmetric key is used as authorization for an entity to access the confidential data. The establishment of shared symmetric keys – and consequently the authentication of the involved parties – is normally done with key management protocols which employ public-key primitives.

Symmetric-key ciphers can be classified into block ciphers and stream ciphers. Block ciphers encrypt fixed-size plaintext blocks and deliver a ciphertext block of the same size. For a given key, a unique plaintext block will result in a unique ciphertext block. A block cipher with a specific key can therefore also be regarded as a bijective mapping between plaintext and ciphertext blocks. Stream ciphers on the other hand can encrypt messages of arbitrary size. From a given cipher key, a stream cipher generates a key stream which is subsequently combined with the plaintext (usually by means of the exclusive-or operation). Stream ciphers try to mimic the behavior of the one-time pad (OTP), which uses a unique random key instead of the generated key stream, and which is secure in the information-theoretical sense.

Block ciphers can be used in different modes of operations in order to improve encryption security (e.g. CBC [14]), provide encryption of arbitrary length messages (e.g. CTR [14]), data integrity (e.g. CMAC [15]), or both confidentiality and data integrity (CCM [16], GCM [17]). There also exist modes of operation so that block ciphers can be used as hash functions (e.g. Davies-Meyer [1]) or as key stream generators for stream ciphers (e.g. BMGL [18]). This opens up the possibility of providing several security services with the help of a single block cipher implementation, used in several modes of operation. This approach seems very attractive for memory-constrained devices.

Important block ciphers include the Data Encryption Standard (DES) algorithm and its extensions (e.g. Triple-DES) [19], which has been the most important cryptographic symmetric algorithm for several decades. Due to the limited key size of 56 bit, the U.S. National Institute of Standards and Technology (NIST) has initiated and led an open call and selection process for a replacement algorithm – the Advanced Encryption Standard (AES). This process lasted from 1997 to 2001, when the Rijndael algorithm [20] was finally selected and standardized as the new AES algorithm [21]. From the fifteen original submissions, five came into the final selection round. At the final selection of Rijndael, NIST emphasized that none of the finalists (MARS, RC6, Rijndael, Serpent, Twofish) had been found to have severe weaknesses and are considered to have sound security [22]. Therefore, the AES finalists are worth consideration in special environments like embedded systems; e.g. RC6 has been shown to have very favorable properties on embedded processors under memory constraints [24]. Another interesting block cipher family for embedded devices are the variants of the Tiny Encryption Algorithm (TEA) [23], which are based on very simple and fast operations.

Stream ciphers have originally been intended for hardware implementation and have received not so much attention from the research community as block ciphers. A popular stream cipher is RC4, which has been shown to have weaknesses, but is still widely used today. The European Network of Excellence in Cryptology (ECRYPT NoE) has initiated the ECRYPT Stream Cipher Project (eSTREAM) in order to identify new interesting stream ciphers, suited for either software and/or hardware implementation [25]. This effort has led to the publication and analysis of new interesting stream cipher designs, e.g. Trivium [26].

In networks, there are several different strategies for providing confidentiality. In the network protocol layer stack, security can be integrated into the data-link layer (OSI layer 2) or into higher layers (e.g. network or application layer). As the data-link layer deals with directly connected network nodes, the solutions at this layer are on a hop-to-hop basis. On

the other hand, solutions on the higher layers are able to provide end-to-end security directly. Both approaches have their advantages and drawbacks. In hop-to-hop solutions, network nodes between sender and receiver have access to the sent data and must therefore be trustworthy. Choosing this strategy allows intermediate nodes to process the data en-route to the ultimate destination, which is an important fundamental technique for e.g. WSNs. On the other hand, in end-to-end solutions intermediate nodes cannot access the sent data. Both communication endpoints can send data through an insecure network infrastructure, without disclosing its content. Intermediate nodes cannot read the encrypted payload and are just required to perform routing.

3.2 Data Integrity

The objective of data integrity addresses unauthorized manipulation of data. This encompasses manipulation of data due to technical reasons (network transmissions errors) as well as deliberate manipulation by an active adversary. Just as the assurance of confidentiality, data integrity is inherently linked to authentication – in particular data origin authentication. To illustrate this, imagine a message sent from an entity A to the intended recipient B. If A's message is manipulated en-route (e.g. by a malicious entity E), then this could equally be interpreted as a change of data origin from the source A to the source E. Therefore, if the origin of data can be determined reliably, then this implicitly provides the integrity of the data.

Hash functions, which belong to the category of unkeyed cryptographic primitives, play an important role in the provision of data integrity. These primitives are able to take a potentially large message and deliver a relatively short value (e.g. 160 bit) – the so-called hash value – which can be seen as a fingerprint of the message. One important property of hash functions is that it is computationally infeasible to find any two messages so that their hashing yields the same hash value.

Attaching hash-based fingerprints is sufficient to detect data manipulation due to technical reasons. In practice, sole protection against such data manipulation is rarely done with the use of cryptographic hash functions but rather with simpler functions like checksums (e.g. byte parity, CRC), which are computationally less demanding and thus faster. However, protection from attackers can only be achieved in combination with symmetric-key or public-key primitives or by using hash functions in combination with a key, e.g. HMAC. This is usually done by encryption of the generated hash value, which can provide data origin authentication and therefore also data integrity.

Until recently, the design of cryptographic hash functions has received comparably little research effort. After the publication of new attacks on the important SHA-1 hash algorithm [28] by Wang et al. [29], there has been a growing effort both towards improving the attacks, as well as designing new, more secure algorithms. Recently, NIST has announced that it will perform a public call and selection process for a new hash standard [27], very similar to the selection process of the AES. This process is projected to last until 2012, finishing with the standardization of one or more new hash functions by NIST.

Hash functions are normally rather demanding when implemented in software. This is especially true for microprocessors of small wordsize (e.g. 8 bit). Therefore it might be favorable to minimize the use of hash functions or consider their replacement by block ciphers in appropriate modes of operation.

3.3 Authentication

The third goal of cryptographic protection in embedded P2P Systems is authentication. Authentication is subdivided into two topics: Entity authentication and data origin authentication [1].

Data origin authentication techniques provide assurance of the identity of the generator of data. The necessity can be motivated by the well-known Man-in-the-Middle Attack: When Alice wants to send Bob a message she assures the integrity by an extension with the hash value of the message. An eavesdropper Eve is able not only to watch the communication of Alice and Bob but Eve is also able to replace it by her own message. Since Bob does not receive any information about the author of the message, he is not able to discover the intervention by Eve. If Alice and Bob run a Diffie-Hellman Key-Agreement, Eve can install her own key in the middle of Bob and Alice and the following encrypted communication can be watched by Eve. Authentication primitives ensure that classical Man-in-the-middle attacks can be detected.

It is easy to see, that message authentication can be achieved by only using secret information, i.e. secret keys, which are only known by Alice and Bob. Symmetric as well as asymmetric cryptographic algorithms are used in practice; their choice depends on the conditions on the cryptosystem.

The classical MACs are symmetric cryptographic primitives. The most common MACs are based on hash functions (HMAC) [30]. Like Hash functions, HMACs map a message of arbitrary length to a bit string of a fixed length. The difference to ordinary hash functions is that a secret key influences the results and that the hash function is executed twice. During the first execution the produced key concatenated with the message is compressed and in the second execution the produced key concatenated with the first hash value is transformed to the resulting MAC. There are further constructions of MACs based on cryptographic primitives using symmetric cryptographic keys. For example, in [31] the construction of MACs based on symmetric ciphers is proposed. The MAC is the result of the encryption in CBC mode, which is mapped by a finishing transformation and truncated to the desired length of the MAC.

In both constructions described above, a symmetric key is used, i.e. the generator as well as the verifier of the MAC has to know the key and both have to keep it secret. Therefore a well-designed key establishment protocol has to be implemented. A solution to this problem is to move to asymmetric cryptographic primitives gaining authentication. Digital signatures provide authentication as well as non-repudiation and symmetric keys have not to be agreed by the communicating parties. To confirm the digital signature the receiver uses a public key of the sender, which is certified by a trusted authority. The certificate confirms that the holder of the certificate possesses the according private key.

There are several standardized digital signature algorithms: RSA, DSA and ECDSA [32,33,34]. The first two algorithms RSA and DSA are based on finite groups with a multiplicative structure and therefore long integer arithmetic has to be provided. This is the reason why these methods implicate a large amount of memory to store temporary results. In particular, restricting to devices with limited memory and computational power these conditions might become a bottleneck. Digital signature algorithms based on elliptic curves are better suited for limited devices, because storage can be saved in comparison to RSA and DSA. Furthermore since elliptic curves are defined over finite fields, there is a lot of theoretical margin in the definition of the finite field. Of course, there are some conditions on the finite fields to gain the full security of elliptic curve cryptosystems, but, in particular

the characteristic of the finite fields can be adapted to the register width of the processor. Therefore the architecture of the platform can be used to obtain optimal performance.

In embedded P2P systems several components will communicate via different communication channels. That is why it might be possible, that unauthorized parties try to enter the communication. *Entity authentication* (also called identification) gains assurance of the identity of one party to a second party. With identification unauthorized parties can be detected and dismissed.

To obtain entity authentication, the claimant has to prove his identity to one or more verifiers. This can be based on three different classes [1]:

1. *Knowledge*: The claimant has to confirm the knowledge of an expected secret. The secret may be personal identification numbers (PINs), passwords or public and secret keys.
2. *Possession*: To prove the identity by possession, physical accessories are used, e.g. magnetic-stripe cards, chip cards, or hand-held customized calculators. In such authentication schemes these devices compute time-variant passwords.
3. *Inherence*: Providing authentication by inherence involves the use of human physical characteristics or biometrics.

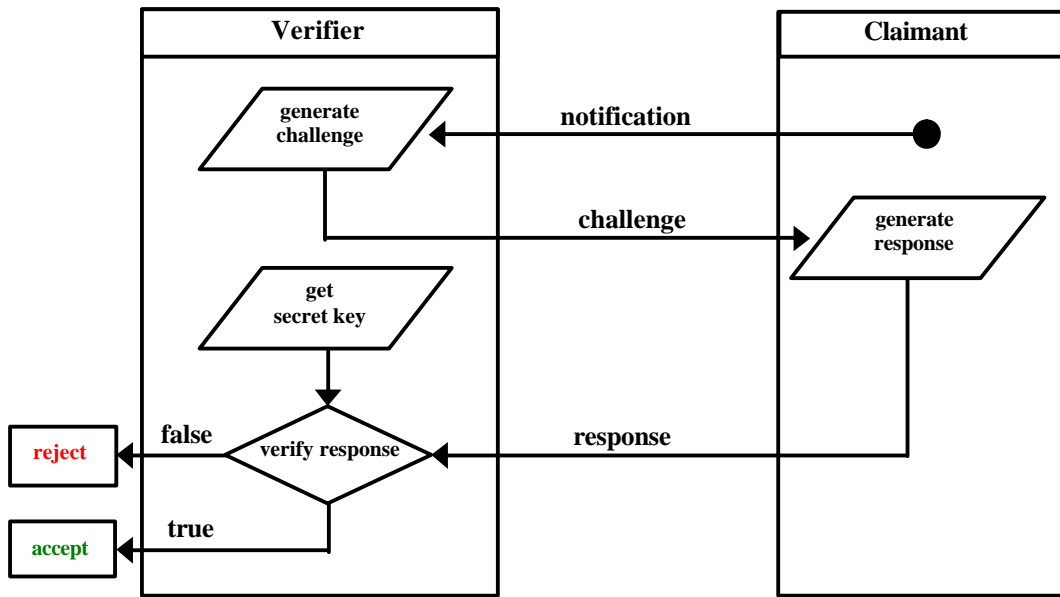
In embedded P2P systems entity authentication has to be based on knowledge of secrets. Identification can be realized using digital signatures or running challenge-response protocols.

Implementing entity authentication by digital signatures, the communication partners have to sign their contributions using their private signature key. The identification of an entity is included in the verification of the digital signature. If the authenticity of the entity cannot be confirmed, then the received data is ignored. Obviously, digital signatures enable asymmetric data flow, which means that the sending party does not need to wait for a response by the verifying one. Authentication based on signatures is useful in systems, where the communication traffic is a bottleneck for its efficiency.

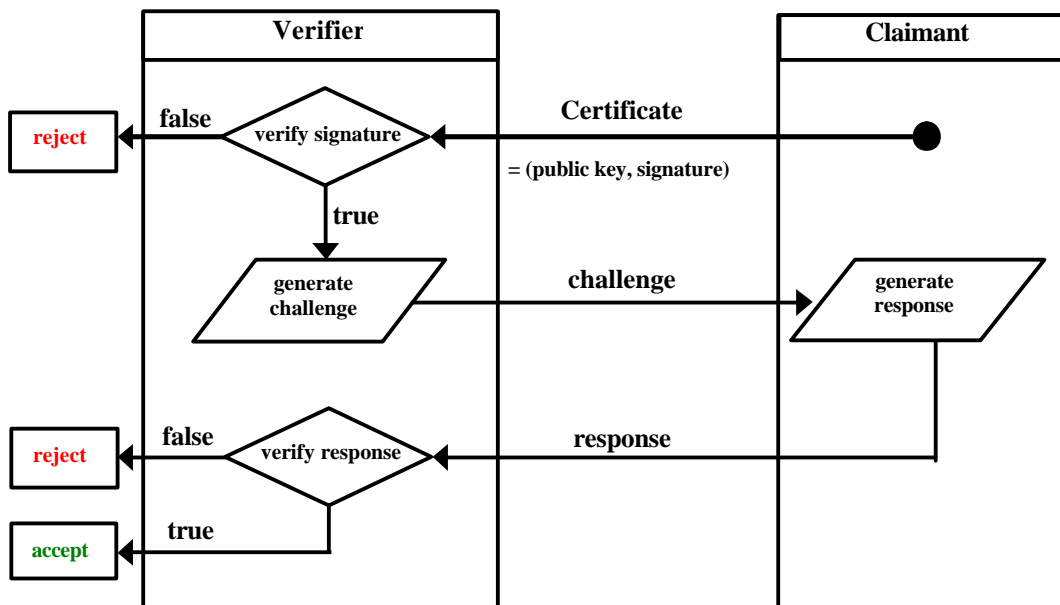
An alternative to digital signatures are so-called challenge-response protocols. The advantage of those protocols is that they are not restricted to asymmetric cryptographic primitives and hence a public-key infrastructure is not mandatory. Challenge-response protocols can be outlined as follows:

1. The party A (claimant) sends a notification to the verifier (B).
2. B sends a nonce to A.
3. A proceeds the nonce involving the secret and returns the response.
4. B verifies the response. If it is correct, the entity authentication is granted.

The complexity of step 4 depends on the underlying cryptographic system. If the challenge-response protocol is based on symmetric cryptographic primitives, the party A for example encrypts the nonce using a secret symmetric key. To verify the response, the verifier has to decrypt the response or, alternatively, has to encrypt the nonce too. All in all, B has to use the same secret key. Therefore the verifier must have access to a centralized data base, where the keys of all possible communication partners are stored. It is obvious, that a high request of security is addressed to the key database and the transport channel. To implement this variant of challenge-response protocol all strong symmetric encryption schemes can be used. The output length of the most common schemes is less than 33 byte. The following diagram describes the work flow of symmetric challenge-response protocols a little bit more in detail:



If the challenge-response protocol is based on asymmetric ciphers, decentralized solutions are feasible. In step 1 the claimant sends his certificate in the notification. If the certificate is valid, the two parties run a one-sided Diffie-Hellman key agreement, such that the verifier uses a random number as private key. The response is the common key.



Using asymmetric ciphers central databases can be avoided, but the traffic between the two parties A and B is higher in comparison to challenge-response protocols based on symmetric ciphers. Caused by the security requirements on asymmetric cryptographic primitives, the length of the challenge and the response is at least 160 bit if elliptic curves are used. In the case of discrete logarithm problems on multiplicative groups the length of the messages increases to more than 1,000 bit.

3.4 Non-Repudiation

Non-repudiation is a stronger form of authentication which allows the sender's identity to be verified by a third trusted party (TTP). The main goal of non-repudiation is to prove that

a message has been sent and received. In addition, non-repudiation can also be used to provide other similar services as one can see in the table below.

A direct conclusion of non-repudiation is that the sender cannot deny sending a particular message. In other words this is like the procedure done by a notar who verifies the signer's identity, witnesses the signing and puts a seal on the document, indicating that the signing has been witnessed.

The International Organization for Standardization (ISO) notes in the draft standard for Guidelines for the Use and Management of Trusted Third Party Services that TTPs may be involved in the provision of non-repudiation services, depending on the mechanisms used and the non-repudiation policy in force.

Non-Repudiation is not possible with a symmetric cryptographic algorithm because in fact more than one person knows the symmetric (secret key). So while it is possible to reduce the number of possible senders, it is not possible to determine the one among the owners of the secret key who sent the message.

In the following table we consider the various forms of "Non-Repudiation Services of a Message" [35]:

Name	Non-repudiation means
Approval	Proof of who is responsible for approval of the content of a message
Sending	Proof of who sent a message
Origin	Proof of who approved and sent a message
Submission	Proof that a delivery authority has accepted a message for transmission
Transport	Proof that a delivery authority has given the message to the intended recipient
Receipt	Proof that the recipient received a message
Knowledge	Proof that the recipient recognized the content of a received message
Delivery	Proof that the recipient received and recognized the content of a message

Modern digital signature schemes like RSA, DSA or ECDSA provide non-repudiation [32,33,34]. In all signature schemes the signer has to keep the complete private signature key to its own. All other parties, like the trusted authority or the verifier of the signature, work with public parameters only. If the parameters of the cryptosystem guarantee its best security, the secret signature key cannot be compromised using the public key or public system parameters.

Application fields, where non-repudiation in everyday's life is very important, are for instance:

- E-commerce,
- Banking networks,

- Legal networks

3.5 References

- [1] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, 1997. ISBN 0-8493-8523-7.
- [2] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band. IEEE Standard 802.11g-2003, June 2003. ISBN 0-7381-3700-6.
- [3] Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs). IEEE Standard 802.15.1-2005, June 2005. ISBN 0-7381-4707-9.
- [4] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Standard 802.15.4-2003, May 2003. ISBN 0-7381-3677-5.
- [5] ZigBee alliance. <http://www.zigbee.org>.
- [6] 3rd Generation Partnership Project (3GPP). <http://www.3gpp.org>.
- [7] William Allen Simpson. The Point-to-Point Protocol (PPP). Internet standard RFC 1661, July 1994. Available online at <http://www.faqs.org/rfcs/rfc1548.html>.
- [8] IPsec protocol family. Specified in Internet standard RFCs 2367, 2403, 2404, 2405, 2410, 2411, 2412, 2451, 2857, 3526, 3706, 3715, 3947, 3948, 4106, 4301, 4302, 4303, 4304, 4305, 4306, 4307, 4308, 4309, 4478, 4543, 4555, 4621, 4806, and 4809. Available online at <http://www.faqs.org/rfcs>.
- [9] Alan O. Freier, Philip Karlton, and Paul C. Kocher. Draft of SSL 3.0 Specification. Available online at <http://wp.netscape.com/eng/ssl3/>.
- [10] Tim Dierks and Eric Rescorla. The TLS Protocol, Version 1.2. Available online at <http://www.ietf.org/html.charters/tls-charter.html>.
- [11] Secure Shell (SSH). Specified in Internet standard RFCs 4250, 4251, 4252, 4253, 4254, 4255, 4256, 4335, 4344, 4345, 4419, 4432, 4462, 4716, and 4819. Available online at <http://www.faqs.org/rfcs>.
- [12] The International PGP Home Page: <http://www.pgpi.org/>.
- [13] Secure / Multipurpose Internet Mail Extensions (S/MIME). Specified in Internet standard RFCs 2311, 2312, 2631, 2634, 2785, 2876, 2984, 3058, 3125, 3183, 3126, 3185, 3217, 3218, 3278, 3274, 3370, 3394, 3114, 3537, 3560, 3565, 3657, 3851, 3850, 3852, 3854, 3855, 4010, 4056, 4134, 4262, and 4490. Available online at <http://www.faqs.org/rfcs>.
- [14] National Institute of Standards and Technology (NIST). Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation, December 2001. Available online at <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>.
- [15] National Institute of Standards and Technology (NIST). Special Publication 800-38B: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005. Available online at http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.
- [16] National Institute of Standards and Technology (NIST). Special Publication 800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, May 2004. Available online at <http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C.pdf>.
- [17] National Institute of Standards and Technology (NIST). Draft Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication. Available online at http://csrc.nist.gov/publications/drafts/Draft-NIST_SP800-38D_Public_Comment.pdf.

- [18] Johan Håstad and Mats Näslund. BMGL: Synchronous key-stream generator with provable security. Primitive submitted to NESSIE, September 2000. Available online at <https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions.html>.
- [19] National Institute of Standards and Technology (NIST). FIPS-46-3: Data Encryption Standard, October 1999. Available online at <http://www.itl.nist.gov/fipspubs/>.
- [20] Joan Daemen and Vincent Rijmen. The Design of Rijndael. Springer Series on Information Security and Cryptography. Springer, 2002. ISBN 3-540-42580-2.
- [21] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001. Available online at <http://www.itl.nist.gov/fipspubs/>.
- [22] AES Round 2 Information: <http://csrc.nist.gov/CryptoToolkit/aes/round2/round2.htm>.
- [23] Roger Needham and David Wheeler. Tea extensions. Technical report, University of Cambridge, 1997. Available online at <http://www.cl.cam.ac.uk/ftp/users/djw3/xtea.ps>.
- [24] Johann Großschädl, Stefan Tillich, Christian Rechberger, Michael Hofmann, and Marcel Medwed. Energy Evaluation of Software Implementations of Block Ciphers under Memory Constraints. In *Design, Automation and Test in Europe (DATE) 2007, 16-20 April 2007, Nice, France, Proceedings*, pages 1110-1115. IEEE Computer Society Press, 2007.
- [25] ECRYPT Stream Cipher Project (eSTREAM): <http://www.ecrypt.eu.org/stream/index.html>.
- [26] Christophe De Cannière and Bart Preneel. Trivium Specifications. Available online at http://www.ecrypt.eu.org/stream/p2ciphers/trivium/trivium_p2.pdf.
- [27] National Institute of Standards and Technology (NIST). Hash Function Workshop. <http://www.csrc.nist.gov/pki/HashWorkshop/index.html>.
- [28] National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at <http://www.itl.nist.gov/fipspubs/>.
- [29] Xiaoyun Wang and Yiqun Lisa Yin and Hongbo Yu. Finding Collisions in the Full SHA-1. In *Victor Shoup, editor, Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of Lecture Notes in Computer Science, pages 17-36. Springer 2005.
- [30] H. Krawczyk, M. Bellare and R. Canetti: HMAC: Keyed-Hashing for Message Authentication, Network Working Group, Request for Comments RFC 2104, 1997.
- [31] International Organization for Standardization: ISO/IEC 9797-1: Information Technology - Security Techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher, 1999.
- [32] RSA Laboratories: PKCS#1 v2.1: RSA Cryptography Standard, 2001
- [33] National Institute of Standards and Technology: FIPS-186-2: Digital Signature Standard (DSS), 2000.
- [34] American National Standard for Financial Services : ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1999.
- [35] V. Bonnet, K. Boudaoud, M. Gagnebin, J. Harms and T. Schultz: Online Dispute Resolution Systems as Web Services, Available online at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=899107.

4 Embedded P2P Systems and Applications

Problems in EP2P networks are very complex and most of them are not solved till this day. In this section, we will discuss dangerous EP2P-threats by the example of typical EP2P applications and will try a careful analysis of all required aspects. For certain threats solutions are known. But in this section, we want to give a unified-, coherent classification of EP2P networks, threats and system requirements.

4.1 Aspects of Embedded P2P Systems

Before giving a closer look to EP2P-specific threats and application requirements, the attempt of a EP2P network classification is given in this section.

4.1.1 P2P Network Types

In general, peer-to-peer networks do not have a clear definition of server- and client task, or a definition of how clients and servers divide up their jobs among themselves. The term P2P rather defines a class of network applications that take advantage of resources (storage, content, context, etc.). With this definition, the problem arise that P2P networks have to find a solution for two disjunctive design philosophies (see Figure 4.1.1):

- The **network centric design** and
- The **application centric design**

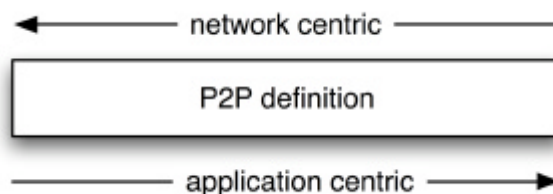


Figure 4.1.1: P2P-domains

While network centric implementations are fully decentralized with non-hierarchical structures and based on symmetric communication, the application centric approach is based on the existence of resources available at the edges of a network without taking into account, how to get access to these resources. An intuitive solution can be a combination of both approaches. But due to disjunctive properties of network- and application centric approaches, we always have to make the best of two bad jobs.

Thus, the P2P system will always be a compromise to co-operate between entities that are regarded to be equal and can require or offer services by acting as decentralized as possible. These compromises lead to the following classification.

One classification of peer-to-peer networks is according to their degree of network-management centralization. While pure P2P networks also have a decentralized peer management, centralized P2P systems are using a central network management server. Hybrid approaches—*on the other hand*—are using centralized servers to keep the information on peers, but the peers are responsible for the hosted information.

Another classification method is based on the structural design. Here we distinguish between structured and unstructured P2P networks, whereas structured as well as unstructured networks can be centralized or decentralized (see Figure 4.1.2).

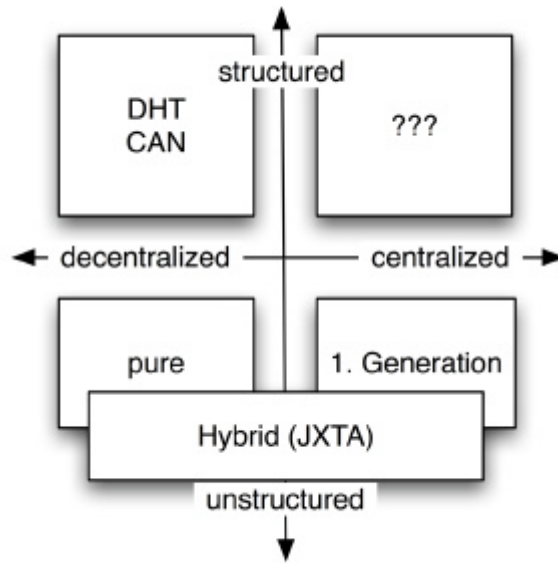


Figure 4.1.2: P2P network classification

Based on this P2P network classification, we can categorize all P2P network generations:

	centralized	de-centralizd	structured	un-structured
1.Generation (unstructured de-centralized)		X		X
2.Generation (unstructured centralized)	X			X
3. Generation Hybrid (centralized- + de-centralized)	X	X		x
3. Generation Structured (DHT)		x	X	
EP2P (structured hybrid)		X	X	x

Table 4.1.2: P2P network classification

X ... mandatory
 x ... optional

4.1.1.1 Centralized P2P network such as Napster

Searching in centralized P2P network is easy due to the existence of a single centralized server, which maintains directories of the shared resources stored on the network. If a user needs the access to a specific file or content, the centralized server creates a list of matching results for a particular search request. The matching mechanism is a simple check, which of the nodes—*claiming to hold the requested file*— is currently connected to the network. The centralized server then offers a list of possible hosts to the requesting user. The user then can choose from the list of nodes where he wants to get the requested information from.

Napster—*for instance*—is based on this centralized and indexed system. Relations between indices of all peers and resources, which are stored on these nodes, are maintained by the

main server. In centralized networks, queries are sent directly to this server, to receive a list of nodes where the requested info can be found.

One problem of a centralized indexes and repositories system are bandwidth- and hardware requirements to support large P2P networks. Another problem is the centralized server which forms a single point of failure. If this single server fails, it can bring down the whole P2P network if no backup mechanisms do exist. In the case of Napster, a cluster of servers is responsible for all queries. If a single server fails, one of the remaining servers will continue to support the network.

Recapitulating all important aspects, we can say, that centralized P2P networks are easy to manage, but also have some serious drawbacks:

- Scalability
- Vulnerability
- Election of servers
- Adequate provision of server
- Consistency

4.1.1.2 Decentralized P2P network such as KaZaA

The concept of decentralization is to remove the central structure of a network such that each peer can communicate equally to all other nodes in the network. When a peer A wants to send a query request to the network, A announces the fact that A is alive to the whole network. B—a *direct neighbour of A*—receives this message and announces the fact that A is alive to all directly connected hosts (C, D, E and F for instance) and so forth. C, D, E, and F are repeating this message in turn.

As soon as A has announced that A is alive, it can send search requests to the network. In a first step, this message is directly sent to B, which forwards the message to C, D, E and F. If for example D has the information which is requested by A, D sends a copy of this information back to B which forwards the response to A.

In decentralized hash table networks, each file stored with in the system is given a unique id, typically a hash of its content, which is used to identify a resource. Given this unique ID, a resource can be located quickly almost independent of the size of the network. But this mechanism implicates the problem that it is impossible to perform keyword search within the network. If a peer is looking for a file from another peer, it must obtain this key first in order to retrieve the file. Another problem is that these systems are also susceptible to malicious activity by untrustworthy nodes. They can discard a query, insert large amount of data, or flood the network with queries to degrade the overall performance.

4.1.1.3 Structured P2P network such as CAN

In [1] a structured P2P system is based on the fact that the location of an object (resource) can always be determined by a globally agreed-upon scheme, e.g., hashing the resource's key. If someone knows the key for a desired object, one can easily find the location where that object should be. If the desired object exists several times in the network - for instance, the network hold many copies of a given file – the network has to decide which object (file) should be associated with the given key.

CAN—*for instance*—is a distributed and decentralized P2P network structure based on hash tables. Thus, CAN is similar to other DHTs, but CAN was designed to be scalable, fault tolerant, and self-organizing.

The structure of CAN is built around a virtual multi-dimensional Cartesian coordinate space in which the entire coordinate space is dynamically partitioned among all the peers in the system such that every peer possesses its individual, distinct zone within the overall space. For routing, all peers maintain Routing tables holding virtual coordinates and IP addresses of all neighbours. To find the best routing decision, a simple Greedy-algorithm can be used in this structured environment.

4.1.1.4 Unstructured P2P network such as Gnutella

In unstructured P2P networks, we neither have a centralized directory nor any precise control over the network topology nor file placement itself. Gnutella is an example of such designs. Loosely coupled nodes form the whole network. Thus, the placement of files cannot be based on any knowledge of the topology (as it is in structured designs).

To find a file, a node queries all its neighbours by flooding, where the query is propagated to all neighbours within a certain radius. These unstructured designs are extremely reliable to nodes entering and leaving the system very frequently. But unfortunately, current search mechanisms are extremely un-scalable, and are often generating large additional loads on the network.

4.1.1.5 Hybrid P2P network such as JXTA

- Has a central server that keeps information on peers and responds to requests for that information.
- Peers are responsible for hosting available resources (as the central server does not have them), for letting the central server know what resources they want to share, and for making its shareable resources available to peers that request it.
- Route terminals are used addresses, which are referenced by a set of indices to obtain an absolute address.

4.1.2 Peer capabilities and environment constraints

While P2P networks are classified as described above, these networks can be used in different environments and applications imposing different constraints on each class of P2P network. We have distinguished between the three different constraint domains. The first domain covers constraints resulting from the computing platforms which form a P2P network. The second domain groups constraints which are imposed by the type of communication and network management used. The third constraint domain results from the possibility of nodes in P2P network to move and their inherent mobility patterns.

4.1.2.1 Platform Resources

This classification is based on the resource capabilities of nodes participating in P2P networks. In principle, we can distinguish between 3 basic device classes:

- Highly Constrained Devices (WSN nodes, RFID tags)
- PDAs, Mobile Phones, etc.

- Personal Computers, servers (workstation class)

Each of these device classes has different requirements on computing, storage and energy resources which are grouped into one group called performance. There are many nuances possible between the described three classes but in general these can be used as representatives. In Figure 4.1.3 this relation is illustrated using the previously defined classes.

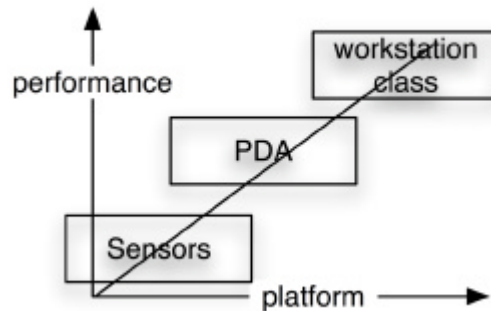


Figure 4.1.3: Platform resource classes

The first class with the lowest performance and the most restrictions on the resource requirements is the class of *highly constrained devices*. One of the best known and researched representatives of this class are the wireless sensor nodes (WSNs). Wireless sensor nodes have very limited computational power and memory resources. They also have only very limited battery power which must be used efficiently in order to achieve a lifespan of several years. RFIDs can also be counted to this class of devices.

The next class in this domain are the PDA style devices. This class is the most heterogeneous one containing platforms like PDAs, mobile phones and other embedded devices. The devices in this class have much more computational power than the devices in the previous class and but have also restrictions concerning power and memory. Although the devices are not as powerful as the devices of the next class they can host almost the same applications.

The last class in this domain represents the workstation type of devices. There exist no predominant limitations for the devices of the workstation class. Although they can vary in their characteristics depending on the use case (laptop, workstation, server, etc.).

4.1.2.2 Communication Facilities

This classification is based on the underlying network structure or concept. In this classification, we define the availability and characteristics of network links and the way how to structure the resulting network. Therefore we can distinguish between peers using the following parameters:

- Wireless – wired
- Self-organizing – organized

The first parameter classifies the peers according to their communication link qualities. We distinguish between wired or wireless communication links. Wired means that there exist preconfigured wires which are used to connect the peers in some way. But it doesn't determine the resulting type of network topology or organization which will be formed by these peers.

The second parameter classifies the peers or more precisely the network management capabilities of the peers according to their organization process. We distinguish between organized and self-organizing networks. In a self-organizing network there exist no preconfigured paths and routes as well as the topology and organization is unspecified. All of these parameters are decided at generation and modified constantly in order to achieve the most efficient network at each time. In a organized network all (or the most) of the previously mentioned parameters are specified from the start and do not change over time. In Figure 4.1.4 common network representatives according to their communication link qualities in relation to the network organization are illustrated.

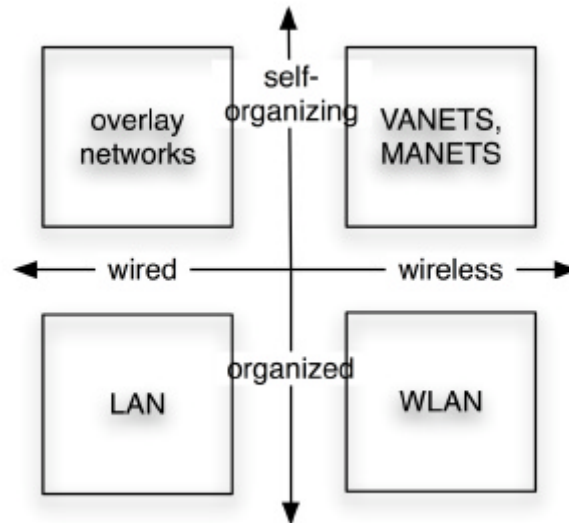


Figure 4.1.4: Communication and networking types

The not comprehensive list of representatives in Figure 4.1.4 are shown to clarify the classification based the described parameters. There are also several hybrid approaches possible which can't be classified by one specific parameter.

4.1.2.3 Device Mobility

This classification is based on the node mobility as well as their movement and online patterns. While mobile nodes can move absolute undeterministic and also join and leave the P2P network at unpredictable times, we also have node classes which move in a predefined manner or participate in P2P networks at fixed (predefined) times. Therefore we can distinguish between peers based on the following parameters:

- Mobile – static
- Non-deterministic – deterministic mobility

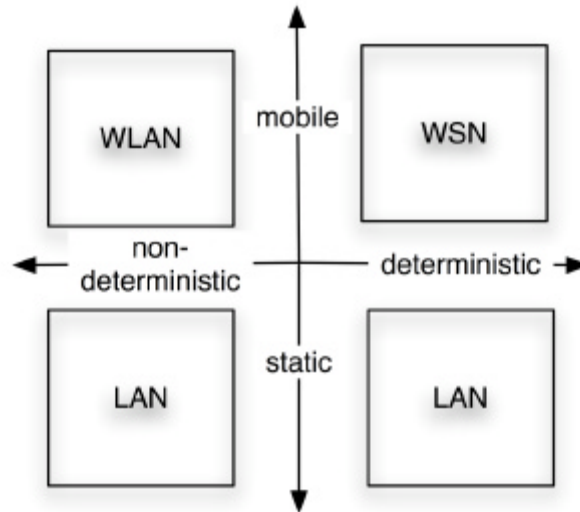


Figure 4.1.5: Device mobility classes

The first parameter classifies the mobility of peers participating in a P2P network. On the one end of the scale there are the peers which mobility is *static*. This means that these hosts will not move from a specified location to another whether they have the mobility to move or not. On the other end of the scale there are peers which mobility is *mobile* are located. Mobile peers can move from one location to another.

The second parameter has two meanings. The first meaning and most relevant one specifies the online pattern of the peer. On the one end of the scale the non-deterministic peers or networks can be found. This means that it is not possible to know at what time or for how long a specific peer or part of a network is online (participate in the network). On the other end of the scale the deterministic peers or networks are located. These peers have a specified online duration and/or a specified connection period. The second meaning is that also the movement pattern of a peer can be specified. Therefore this would specify that the non-deterministic peers would move in a chaotic manner and the deterministic peers move in a prespecified way. In Figure 4.1.5 some distinct representatives for the possible combinations are illustrated.

4.1.2.4 Possible Combinations

	Constrained devices	PDA class	Workstation class	wireless	wired	self organizing	organized	mobile	static	non-deterministic	deterministic
1. Generation (unstructured decentralized)	!	?	X	x	X	X	!	?	X	X	X
2. Generation (unstructured centralized)	!	?	X	x	X	!	X	?	X	X	X
3. Generation Hybrid (centralized + de-centralized)	!	?	X	x	X	X	X	?	X	X	X
3. Generation Structured (DHT)	!	?	X	x	X	x	X	?	X	X	X
EP2P (structured hybrid)	X	X	x	X	X	X	x	X	x	X	x

Table 4.1.2.4 Combinations of network types and aspects

X ... mandatory
 x ... optional
 ? ... not necessarily applicable
 ! ... not applicable

As seen in table 4.1.2.4, the **concept of SMEPP** is very ambitious. While **EP2P** networks are still focused at **wired** and **non-deterministic** networks, novel application-requirements come to the fore due to the efficiency of these days embedded applications. **Mobile-** and **wireless** communications between **low-performing devices**, but also the interaction with **infrastructured** networks creates new demands on security primitives, -protocols and –management.

4.2 Threats

In an EP2P system, it is especially challenging to develop a secure environment as it introduces a new paradigm for distributed computing where the traditional roles of client and server no longer exist. While the most widely used distributed computation models are based on the asymmetric connection of client and server, often involving pre-existing infrastructures, the EP2P scenario features a symmetrical network element where communication takes place in a **dynamic** and **ad-hoc** fashion. Since the EP2P systems are resource and cost constraints (e.g. lack of tamper-resistant packaging) and the general susceptibility of the wireless communication channels to attacks, it is extremely vulnerable against both internal and external attacks by adversaries.

In the following sub-sections, we will identify some of the threats in a EP2P system.

4.2.1 P2P Network Threats

In the traditional networks, the security services are provided by end-to-end mechanisms such as SSH, SSL and etc. However, since the dominant traffic pattern in EP2P is many-to-one, and there is a need of in-network processing of data for e.g. data aggregation and duplicate elimination, it is difficult to implement the end-to-end security mechanisms in the EP2P.

In the following sub-sections, we will discussed the identified EP2P threats categorized as follows:

- Threats due to restricted Resources: e.g. Exhaustion of power, e.g., hello flooding and acknowledgement spoofing
- Threats due to Communication Facilities: Attack on routing protocols, e.g., spoofed, altered or replay of routing information
- Threats due to the Device Mobility: e.g. Attack on data aggregation, cheating with position, speed and identity, colaboration attacks
- Other Threats: Side channel attacks, Node Compromise, Impersonation Attack

4.2.1.1 Threats due to Restricted Resources

In a EP2P environment, most of the device are embedded deivces. These embedded devices are often limited in their resources such as battery life, processor power and so on. As a result, they are often subjected to attacks targeting on their vulnerabilities on having limited resources. Adversaries can either launched a *mote-class attack*, using devices

comparable to the embedded devices in the EP2P setup or a *laptop-class attack*, using more powerful devices.

A mote-class attacker launches its attack by having access to a few low-cost sensor nodes (motes) of similar capabilities, whereas a laptop-class attacker can directly launched an attack using a laptop-class node with high computing capabilities, better battery life, higher power and more sensitive communication equipment. A laptop-class attacker can inflict more damages than an ordinary mote-class attacker. For example, a laptop-class attacker is able to jam the radio links of an entire WSN in contrast to the mote-class attacker, who can only jam the radio-links of the neighbouring nodes. A single laptop-class attacker can eavesdrop on an entire WSN and having access to high-bandwidth and low-latency radio links, which allows the attacker to mount and coordinate the attacks. A similar distinction can also be made between the *insider* and the *outsider* attacks.

With to the nature of the limited resources found on the EP2P devices, the attacker can launched a Denial of Service (DoS) attack targeting to incapacitate the device or the entire network. Normally, DoS attacks involve three parameters: Users, a shared service or resource and a maximum waiting time. During a DoS attack, a user is made to wait longer than the predefined maximum waiting time by a malicious user for the use of shared service(s) or resource(s).

A DoS attack on a EP2P network may take several forms, for example:

- *Denial-of-message attack* in which a set of nodes act maliciously and prevent broadcast messages from reaching certain section(s) of the sensor network.
- *Exhaustion of power* in which an attacker repeatedly requests packets from sensors in order to deplete their battery. E.g. hello flooding and acknowledgement spoofing.

4.2.1.2 Communication Facilities Threats

Communication in a EP2P network is mainly done through the wireless medium, which is broadcast by nature. Therefore it is easily subjected to data injection, modification, interception, and replication. An adversary can easily capture the transmitted data by sniffing out the messages that are sent over the air by the embedded devices. With the captured data, the adversary can then deduce valuable information or events with respect to the monitored area.

The following are the common attacks that can be launched by the adversaries:

- **Eavesdropping** – As the nature of EP2P system transmission medium is broadcast over the air, an adversary can easily retrieve valuable data from the transmitted packets that are sent in a EP2P system by sniffing out packets at a particular frequency using commonly available software similar to Ethereal. A customary solution used to prevent eavesdropping is through the use of encryption for data confidentiality.
- **Message Modification** – An adversary can simply intercept and modify the unsecured packets' content meant for the base station or intermediate nodes to disrupt the sensor's value of a particular sensing region. Securing messages against illegal modification can be done using MACs or digital signatures.
- **Message Replay** – An adversary performs a replay attack by first intercepting a valid critical transaction data packet and then re-transmitting it at a later time. This critical transaction data can be, for example, a proof of identity in a form of a response to a challenge sent by a verifier. By re-transmitting the correct response that has been captured earlier to the same challenge, issued by the verifier, an adversary can fool the verifier to believe that the adversary is the valid party in response to the challenge that

was sent out. Message replay can be avoided through the use of a session token, a security nonce or a message timestamp.

- **Message Injection** – This is an attack where an adversary sends out false data into the EP2P network with the objectives of corrupting the collected data, disrupting the routing table, etc. The adversary usually masquerades as one of the node to avoid being detected, and can perform a laptop-class attack which allows it to send multiple copies of the forged message throughout the network to disrupt data aggregation, routing protocols, etc. The solution for securing messages against illegal modification (i.e. MAC or digital signature) can also be used for detection of false data injection.

Although simple and practical solutions have been implemented to prevent most of these attacks in the wireless network domain, these solutions are not feasible to be implemented in EP2P systems due to the limited capabilities of the devices and the topology of the network deployment. This is because current techniques used in wireless networks often increase the overhead of the message sent, thus resulting in a lower throughput for the embedded devices. An increase in message overhead would also imply an increase in the power consumption for sensors due to the extra energy needed to transmit the larger message overhead. As the topology of the EP2P systems ranges from an ad-hoc network to a multi-hop many-to-one network deployment, current end-to-end solutions will not be feasible especially for protocols which require intermediate nodes to aggregate and consolidate the data before forwarding them to centralised sink node.

4.2.1.3 Device Mobility Threats

In the general EP2P network, there are a number of nodes communicating with one or more super-node. This setting can be viewed as an event-based system where the super-node act as *sinks* which subscribe to specific data streams by expressing interest and queries and the nodes act as sources to report environmental events to the sink. The sink is often assumed to be powerful enough to perform computationally intensive cryptographic operations while the sensor nodes have constrained resources in terms of computation, memory and battery power. Usually the networking among sensor nodes is assumed to be highly *ad-hoc* in a sense that the network topology may change rapidly and unpredictably.

Among the sensor nodes in a WSN, there can be some special nodes called *aggregators* that conduct some computational operations on the data from their child nodes, for example, taking the sum, average, maximum or minimum of the data [4]. The resulting data from the aggregators are forwarded to the sinks.

There are two main security threats to the secure data aggregation in WSNs. One is *eavesdropping*, by which an attacker tries to obtain information about the transmitted data between sensing nodes (non-aggregators), between aggregators, and between aggregators and sink. This causes a great damage especially when the data aggregated are of critical importance. The other threat is *forging*, which makes it possible for an attacker to alter the aggregated data or other related information in such a way that invalid aggregated data are accepted as correct or vice versa.

Public-key cryptography will bring great simplicity and efficiency in providing essential security services for data aggregation (in both the hop-by-hop and end-to-end settings) as well as other security services for WSNs in general [9]. Public-key cryptography can overcome the problems of 1-key and $(N-1)$ -key solutions more easily and simply than those solutions based on key pre-distribution.

An interesting research topic would be to construct a *light-weight homomorphic encryption* scheme that can be employed to provide security for data aggregation in the end-to-end

setting. Another interesting topic would be to provide efficient data integrity mechanism for data aggregation in the end-to-end setting, which has not been discussed in the literature sufficiently.

With mobility of embedded devices, the routing path are obtained through a de-centralised manner. Since the routing are not done by a central party, it is subjected to being attack by the adversaries. The adversaries are able to launched the following specific attacks due to the nature of the de-centralise routing protocol in an EP2P network:

- **Spoofed Routing Information** – The adversaries want to cripple a EP2P network by creating routing loops, partitioning the network etc.
- **Selective Forwarding** – A malicious node selectively forwards messages depending on some criteria; thus the attack is less easy being detected by neighboring nodes.
- **Sinkhole Attack** – An adversary attracts all the traffic of a particular area through a malicious node with false routing information and then tampers with the messages passing through it. Mounting a *sinkhole attack* is particularly easy due to the nature of communication in WSN where most of the packets share the same final destination, namely the base station. Thus a compromised node only needs to declare a high quality link to the base station which may lure a large number of nodes to send packets through the compromised node.
- **Sybil Attack** – A single node takes on multiple identities to deceive other nodes. It can reduce effectiveness of a fault-tolerant system which deploys resources redundantly. It can also affect the functioning of geographic routing protocols.
- **Wormhole Attack** – An attacker captures message bits at one location and replays them in another location. A typical *wormhole attack* involves two distant compromised nodes and they falsely understate their distance using some high bandwidth channel available only to them. Well placed wormholes can significantly alter the routing paths to their advantage. Most of the wormhole attacks are mounted in combination with *selective forwarding*, *Sybil attack* or *eavesdropping*.
- **Hello Flood Attack** – Motes send HELLO packets to their neighbors during network setup or neighborhood discovery. Motes which receive these HELLO packets consider the other node to be their neighbor which is within the normal radio range. However, this can be false as a laptop-class attacker with strong transmission power can send HELLO packets to the entire WSN and can advertise a very good link to the base station. Many motes may consider the adversary as their neighbor but nodes far away from the attacker may actually send their packets into oblivion. Thus the network performance will degrade considerably.
- **Acknowledgment Spoofing** – It is a technique used by an adversary to mount attacks on those networks where routing schemes use link-layer acknowledgments to decide the link reliability. An adversary may be able to convince nodes that a weak link is strong or reinforce a dead link by acknowledgment spoofing. Consequently, messages sent via these routes are lost. This attack can also be used to mount several other attacks.

The attacks discussed pose as difficult challenges to the designers of the secure routing protocols. Some of the countermeasures incorporated into the routing protocols after the completion of the design process may not work well. Thus the challenge is to design secure routing protocols from scratch which are resistant to these attacks.

4.2.2 Other Threats

The side channel attack, node compromise and impersonation attacks are 3 of the threats that are identified in this sub-section. As the embedded devices are often deployed in an

unsecure area, they are easily subjected to being tampered by malicious attackers. A malicious attacker can also easily masquerade as an legitimate node since in an EP2P system, a node can easily join and leave the network.

The side-channel attack is a particularly powerful subclass of the implementation attacks, where the adversary exploits the leakage of the secret key material through certain physical properties (side channels) of the device during cryptographic operations. Some of the typical side channels are the execution time [19], the power consumption [20], and the electromagnetic emanation [21],[22]. Side-channel attacks are passive (working environment of device is largely left unchanged) and non-invasive (device is not manipulated) [23] and can be even executed remotely in specific settings [24]. Therefore, side-channel attacks might be hard to detect.

Additionally there is another class of attacks – the so-called implementation attacks – which threatens the security of cryptographic devices. Implementation attacks try to extract the secret keys from the devices, which – if successful – weakens or destroys cryptographic protection.

Side-channel analysis countermeasures on a single device can only increase the workload of an attacker, but can never fully prevent attacks. It is important to defend against such attacks on multiple levels ranging from side-channel resistant implementation of cryptographic algorithms over appropriate cryptographic protocols to a system-wide security policy which allows for graceful degradation of overall security in the case of key compromise of some of the EP2P devices.

Another threat to the EP2P system is node compromise. As the embedded devices are not tamper resistant, it can be easily be compromised by an attacker. When a node is compromised, the attacker is able to retrieve critical information, such as the security keys used in securing the communication or information pertaining to the routing protocols. Using the retrieved information, the attacker will be able to eavesdrop on the communication data or launch other malicious attacks on the EP2P network. Using the retrieved data, various attacks can be launched from compromised nodes.

- **Eavesdropping** – Since an attacker is able to retrieve the security keys from the mote, the attacker is able to eavesdrop on the on-going traffic and sniff out important information that is sent across the sensor network.
- **Node Replication** – With the capturing of the sensor node, the attacker is able to replicate the node at different locations since the attacker has the sensor's program flash, EEPROM and SRAM images.
- **Node Masquerading** – Using the compromised security keys and other information, an attacker can masquerade as the sensor node using a more powerful device such as a laptop. This increase in the capability of the 'sensor' allows the attacker to launch attacks that are more computational intensive.
- **False Injection of Data** – Since the attacker has the control over the compromised nodes, the attacker is able to modify its code to perform false data injection to the wireless sensor network.
- **Denial of Service** – Since the attacker has the control over the compromised nodes, the attacker can simply launch DoS attacks easily either from the node itself or from a laptop masquerading as one of the sensor nodes.

Compromising of the sensor nodes in the WSN, a representative system of EP2P, can be categorized into two different groups according to Becher et al. [2]. The first group of attacks is based on the amount of control and information gained from the captured node. Examples of the attacks in this category are as follows:

- Gaining complete read/write access
- Reading out RAM or flash memory
- Influencing sensor reading
- Manipulating radio communication

The second category of attacks is grouped based on the time needed to compromise the sensor. Becher et al. further divided the group into 3 subgroups [2]. The subgroups are short (< 5 minutes), medium (< 30 minutes) and long attacks (> 30 minutes). It is mentioned in the paper that long attacks can only be carried out in a laboratory using specialized and precise equipment.

It has been shown by Hartung et al. [3] that using the Joint Test Action Group (JTAG) interface or the programming interface found in most of the sensor nodes currently available, an attacker is able to retrieve data stored in both the ROM and the RAM within a time frame of 1 minute. With some analysis done on the data retrieved, the attacker is able to establish the secret keys used in either the encryption or the signing of the data packets, the routing protocols and other sensitive information that are stored in the node itself.

Other than the attacks mentioned by Hartung et al. [3], invasive and non-invasive attacks also allow an attacker to retrieve information from the sensor node's storage media. A non-invasive attack is defined as an attack where physical measurements are taken on the hardware device without any form of structural modification done to the device itself. An attacker can use non-invasive attacks such as optical scrutiny to read out the memory's content. An invasive physical attack is defined as an attack where the attacker physically breaks into the hardware by modifying its structure. Examples of invasive attacks are:

- Focused Ion Beam
- Chemical Etching of protective layer
- Drilling a hole in the storage media and using a microprobe

Such attacks can be used to attack sensors that do not have any interface port. Information is retrieved from the sensor by taking measurements on the memory itself.

Therefore it is a challenge to provide a low-cost software-based solution, that will be able to protect the sensors from revealing its sensitive information to the attackers when they are captured.

In an impersonation attack, a malicious node impersonates a legitimate node. The malicious node then uses the identity of a legitimate node to mount an attack. In the field of wireless sensor networks, the impersonation attack is the initial step which enables the attacker to conduct a range of malicious attacks like eavesdropping, message injection and wormhole attacks. All of these attacks except eavesdropping are active attacks.

Most of the time, cryptographic secrets from captured nodes are used to steal the identity and mount the attack. Several classes of attacks like node replication and Sybil attack can be categorized as an impersonation attack. DoS attack, false message injection, etc. can be mounted using the stolen identity. If more than one stolen identity is available, they can collaborate and mount the attack to incapacitate the original network. Consider the following scenario where an adversary uses a captured identity in addition to nodes deployed by him to form a malicious network. Stolen identities are used as the access point to mount the attack on the original network. Malicious nodes having stolen identities thus become part of both networks and facilitate the attack. Adversaries can also re-distribute these stolen identities to different nodes in the malicious network. Thus stolen identities

can be used by multiple nodes to mount the attack. These types of arrangement can be used to mount DoS attacks, message injection attacks, wormhole attacks, etc.

WSNs operate in a unique environment when compared to normal networks. The main challenge for WSNs is that the network consists of sensor nodes which have limited resources. Sensor nodes have low processing power and storage capability but the main restriction in the case of authentication is the energy consumption when transmitting data. It is important to reduce the amount of data transmitted and received by nodes during the authentication process. The restriction on communication is the largest influence on the authentication mechanisms designed for WSNs. A WSN is likely to contain many nodes. This presents a large target for attackers. Centralized authentication schemes for WSNs which generate a large amount of traffic are therefore discouraged.

4.3 References

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "A Survey on Sensor Networks". IEEE Communications Magazine, 40(8):102–114, August 2002.
- [2] A. Becher, Z. Benenson, M. Dornseif. "Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks". 3rd International Conference on Security in Pervasive Computing (SPC'06), April 2006.
- [3] C. Hartung, J. Balasalle, and R. Han. "Node Compromise in Sensor Networks: The Need for Secure Systems". Technical Report CU-CS-990-05, Department of Computer Science, University of Colorado, Boulder, Jan 2005.
- [4] B. Przydatek, D. Song, and A. Perrig. "SIA: Secure Information Aggregation in Sensor Networks". ACM SenSys '03, 2003.
- [5] B. Krishnamachari, D. Estrin and S. Wicker. "The Impact of Data Aggregation in Wireless Sensor Networks". International Conference on Distributed Computing Systems (ICDCS'02), pp. 575-578, 2002.
- [6] Y. Sang, H. Shen, Y. Inoguchi, Y. Tan, and N. Xiong. "Secure Data Aggregation in Wireless Sensor Networks: A Survey". 7th IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06), 2006.
- [7] D. Wagner. "Resilient Aggregation in Sensor Networks". ACM SASN'04, Washington DC, pp. 78-87, 2004.
- [8] R. Watro, D. Kong, S. Fen Cuti, C. Gardiner, C. Lynn, and P. Kruus. "TinyPK: Securing Sensor Networks with Public Key Technology". ACM SASN'04, pp. 59-64, 2004.
- [9] G. Gaubatz, J.-P. Kaps, E. Oztruk, and B. Sunar. "State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks". 2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec'05), 2005.
- [10] J. Newsome, E. Shi, D. Song, and A. Perrig. "The Sybil Attack in Sensor Networks: Analysis & Defenses", IPSN '04, Berkeley, California, USA, April 2004.
- [11] L. Eschenauer and V. D. Gligor. "A Key-Management Scheme for Distributed Sensor Networks". 9th ACM Conference on Computer and Communications Security (CCS '02), Washington, DC, USA, November 2002.
- [12] A. Perrig. "The BiBa One-Time Signature and Broadcast Authentication Protocol". 8th ACM Conference on Computer and Communications Security (CCS '01), Philadelphia, PA, USA, November 2001.
- [13] S. Capkun and J. Hubaux. "Secure Positioning of Wireless Devices with Application to Sensor Networks". IEEE INFOCOM 2005, pp. 1917--1928.
- [14] L. Lazos and R. Poovendran. "SeRLoc: Secure Range-Independent Localization for Wireless Sensor Networks". ACM Workshop on Wireless Security (WiSe'04), Philadelphia, PA, USA, 2004.

- [15] M. Anand, Z. G. Ives, and I. Lee. “*Quantifying Eavesdropping Vulnerability in Sensor Networks*”. 2nd International VLDB Workshop on Data Management for Sensor Networks, (DMSN'05), Trondheim, Norway, August 2005.
- [16] C. Karlof, N. Sastry, and D. Wagner, “*Tinysec: A Link-layer Security Architecture for Wireless Sensor Networks*”. SenSys'04, 2004.
- [17] D. Wood and J. A. Stankovic. “*Denial of Service in Sensor Networks*”. IEEE Computer, Vol.35, No. 10, 2002.
- [18] C. Karlof and D. Wagner. “*Secure Routing in Wireless Sensor Networks: Attacks and Countermeasure*”. Ad-Hoc Networks, Vol-1, Issues:2-3, pp. 293-315, Elsevier, Sept 2003.
- [19] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Neal Koblitz, editor, Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, number 1109 in Lecture Notes in Computer Science, pages 104–113. Springer, 1996.
- [20] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Michael Wiener, editor, Advances in Cryptology - CRYPTO'99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of Lecture Notes in Computer Science, pages 388–397. Springer, 1999.
- [21] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *Çetin Kaya Koç, David Naccache, and Christof Paar, editors, Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of Lecture Notes in Computer Science, pages 251–261. Springer, 2001.
- [22] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Isabelle Attali and Thomas P. Jensen, editors, Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of Lecture Notes in Computer Science, pages 200–210. Springer, 2001.
- [23] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power Analysis Attacks – Revealing the Secrets of Smart Cards. ADIS Book Series. Springer, 2007. ISBN 0-387-30857-1.
- [24] David Brumley and Dan Boneh. Remote timing attacks are practical. In *Proceedings of the 12th USENIX Security Symposium*, pages 1-14, August 2003.

5 Protection of EP2P Systems

Data origin authentication is a general and necessary issue in the field of protecting P2P- and overlay networks. On the one hand, we need authentication to protect networks against common impersonation attacks (like node replication or message injection). And on the other hand, we need authentication to prevent attacks on routing protocols. Actually, both threats are inherently based on DoS¹ attacks and can therefore be solved by reliable authentication mechanisms.

But due to the fact that the main focus of this section is devoted to securing embedded P2P (EP2P) systems, we have to turn our attention to power- and memory constrained environments. One reason for this is based on the fact that computationally expensive methods cannot be used on such devices. Another reason is the incompatibility between hierarchically organized public key infrastructures (PKI) and the self-organizing character of ad-hoc or sensor networks. Under these certain conditions, it is self-evident that classical authentication mechanisms cannot be used on EP2P systems.

In the case of P2P routing, data origin authentication needs to be extended to multi- or anycast authentication, to reflect group-oriented mechanisms, which are needed to form clusters, support group-oriented applications or apply policies to a group of nodes. Hence, group-oriented data origin authentication is probably the most important component in secure EP2P networks.

Another requirement (beside constrained resources and group-oriented authentication) is the need for an absolutely scalable solution. Thus, a group-oriented authentication mechanism must take into account scalability and efficiency of available cryptographic schemes, since special security primitives can only be used in suitable environments. But all these heterogeneous platforms have to act as a single unit.

5.1 Data Origin Authentication

Data origin authentication is the very fundamental requirement for trusted, reliable and secure communication of distributed entities. Every security and protection algorithm or protocol relies on the authenticity of the involved information. Especially, in dynamic self-organizing networks, like ad-hoc, P2P, or sensor networks, this requirement remains a challenging problem in terms of scalability, efficiency, and performance.

Hashes, MACs and digital signatures are the common cryptographic answers to this daunting challenge. However, these mechanisms have been designed for the traditional non-self-organized network architectures and applying the exact same methods to self-organizing networks results in non-adequate and poor solutions. The inadequateness comes from the amount of possible communication partners, the type of transmitted data and method as well as the underlying physical network layer. These parameters are much more diverse in self-organizing network architectures than in traditional ones.

We now provide an overview of the most promising approaches proposed to provide a solution to this problem. All of them have some limitations which stem from the fact that they have been designed more or less for a specific purpose. Subsequently, we discuss the shortcomings of these approaches in context of embedded P2P networks and outline the additional requirements which must be addressed by an adequate solution.

¹ As described in D4.1 Threat Models for EP2P Networks (Section 2).

5.1.1 Authentication Mechanisms

In order to assure data origin authentication for a given message, the sender and the receiver must share a secret. With this secret the sender can generate an authenticator and the receiver can verify the authenticator. Data origin authentication is guaranteed because *only* the sender and the receiver know the secret. As described in Section 3.3, such a kind of cryptographic system is called symmetric. In other than *unicast* information delivery environments it is not possible to provide authenticity using plain symmetric cryptography since more than one entity possesses the secret key. Common P2P systems are not solely based on unicast transmission but rather use multicast and anycast. Hence, symmetric cryptography cannot be used for authentication. Therefore, to assure data origin authentication the authentication information must be asymmetric. By asymmetric we mean that receivers can verify authentication information but cannot generate it. Authentication with non-repudiation is usually provided by *digital signatures*, an application of public-key cryptography. Since public-key cryptography itself and the problem of key management and certification are *very computational* and *organizational expensive* this solution cannot be used for the designated project environment. Feasible asymmetry based data origin authentication mechanisms, in the context of embedded P2P systems, can be classified according to a survey by Challal et al. [1] into six different categories based on the non-repudiation property, the type of used authentication information asymmetry, and the signature reduction scheme.

5.1.1.1 Secret Information Based Asymmetry

In this category, the sender uses a set of secrets to authenticate a message and gives to each receiver only a partial view of the used secret. This allows the receiver to verify the authenticity of a received message without being able to generate valid authentication information. The following approaches are considered as the best representatives.

The approaches [2-7] are based on a polynomial scheme assuring a k out of n multicast authentication. This means, that any k out of n multi-receivers cannot commit a substitution or impersonation attack on even a single receiver. The authors proved that this protocol is an unconditional secure authentication scheme. This protocol tolerates packet loss and it doesn't introduce latency at the sender or at the receiver. Some improvements for this scheme have been proposed which extended the amount of messages which can be authenticated with each polynomial share.

Other approaches [8] and [9] are based on the assumption that a misbehaving receiver does not have sufficient computational resources to guess secret keys, used to authenticate messages, in reasonable time. Most of them rely on the fact that it is difficult to guess a valid MAC for a message without having the secret key used by legitimate communication parties. In particular it must not be possible for up to w colluding adversaries to know all the keys held by a regular member.

5.1.1.2 Time Based Asymmetry

The approaches belonging to this category are limiting the lifetime of keys used to authenticate packets by associating them with a distinct interval in time. After this distinct interval of time the key used for authentication is disclosed by transmitting to all possible receivers. This means messages authenticated using a distinct key are only valid during a specified time period. If the time period has elapsed messages authenticated using the same key must be rejected by the receivers since the key is not valid anymore. Therefore, an

attacker computing message authentication information on behalf of a valid sender using an already received key doesn't produce a valid authenticated message.

The approaches in this category [10], [11], [12], and [13] are similar. They differ mostly in the amount of keys used during one distinct time interval and other protocol specific properties. But all approaches are tolerant to packet loss since each packet can be authenticated individually. Also the generation and verification of the authentication information is very efficient. Another advantage of this protocol is the applicability in resource constrained environments, since the authors have also proposed a scalable lightweight version [14]. The main disadvantage is the requirement for synchronization between all the involved communication nodes. This especially renders these approaches useless for dynamic environments with high frequency of node leaves/joins and movement of nodes.

5.1.1.3 Hybrid Approaches

The approaches from the hybrid category try to combine the positive and to eliminate the negative aspects of the secret information and time asymmetry approaches. The advantage of the secret information asymmetry approach is that packets are verified as soon as they are received. The main drawback is the sensitiveness to collaboration of misbehaving nodes. In contrast, the advantage of the time asymmetry approach is that collusions are impossible as long as the communication partners are synchronized and the authentication keys are disclosed after their associated time interval has passed. The main drawback is the requirement of buffering packets before verification.

All the approaches in this category are based on one-time signatures [15], [16], and [17]. The sender uses a set of keys to authenticate messages and only a subset of these keys is disclosed along with the packet. This subset of keys allows the verification of the packets' authentication information without being able to generate it. Hence, the authenticity of packets can be verified immediately. If the same set of keys is used to authenticate packets too often, all receivers would acquire the whole set of the sender's keys. In order to prevent this, the sender changes the set of keys periodically. For the key generation a one-way key chain mechanism is used which allows the receivers to verify their authenticity and are able to recover them if some are lost.

5.1.1.4 Signature Propagation

To assure non-repudiation for a given message the sender has to sign it using its private key. Since common digital signature mechanisms are computationally very expensive they are not applicable in an embedded P2P environment. Therefore, alternatives to this traditional solution must be found. The first alternative is to sign only a reduced number of packets and to append information which in turn allows verifying the authenticity of the other packets without a signature. This approach is called *signature propagation* since the single signature effects propagate throughout the packets' relationship. There are several approaches which are based on this mechanism. They can be divided into those which tolerate packet loss and those which don't.

The approaches [18], [19] can be counted to the subcategory which don't tolerate packet loss. The first approach divides a stream of packets into blocks and appends to each block n the hash of the following block $n + 1$. The first block in this sequence is signed and since the hash of the next block in the sequence is included, the authentication information is propagated throughout the rest of the sequence. In approach [20] each data block n is appended with a one-time public key which has been used to one-time sign the following block $n + 1$. Since the first one-time public key is digitally signed the signature is

propagated throughout the stream. The received packets can be verified immediately by both approaches although for the first approach the whole stream must be known in advance.

Packet loss tolerating approaches [11], [20], [21], [22], [23], [24], and [25] also embed authentication information based on hashes in packets. But instead of only embedding it in one packet, the authentication information of a distinct packet is embedded in several packets. The redundancy of authentication information creates a relationship among these packets which is used for estimation on the packet loss probability. Therefore, all of these approaches try to minimize the authentication information redundancy and to maximize packet loss resistance. The approaches either introduce latency at the senders or the receivers or both. The parameters necessary to calculate the required redundancy and relationship topology cannot be obtained easily.

5.1.1.5 Signature Dispersal

Another solution to data origin authentication with non-repudiation is to not only embed authentication information in packets which link them to a single signature but instead adding a block signature and hashes of distinct other packets to each packet. This mechanism is called signature dispersal and is similar to signature propagation.

The approach described in [26] and [27] divides a stream into blocks and signs them block-wise. The digests of the block packets are organized into a tree, for instance a binary tree, whereas the packets form the leaves. The block signature, the packet position in the block and the digests siblings of each node in the packet's path to the tree root are appended to each packet sent. This mechanism allows the individual verification of each packet even if $n - 1$ of n packets are lost. Other approaches [28] and [29], [30] use IDA² to disperse the n hashes of n block packets as well as the block signature into n pieces which can be reconstructed even if $n - m$ packets are lost. All approaches introduce latency at the sender and the receiver. The solutions also include large authentication information into each packet and are computationally more expensive than other alternatives.

5.1.1.6 Differed Signing

The last alternative uses a combination of signing and one-time signing to address non-repudiation. The one-time public keys, which are used to verify the one-time signature of the transmitted packets, are signed with a conventional certified private key. The remaining problem of signing and verifying the one-time public keys without interfering with real-time transmission is addressed by the approaches in this category.

The approaches [31], [32], and [33] sign the keys off-line and the cache in a buffer for future use. The cached keys are then used to authenticate message on-line using one-time or k-time signing, which are known to be fast. The scheme tolerates packet loss and received messages can be verified as they arrive. The main drawback of the proposed approaches is that the receiver has to verify the conventional digital signature and several one-time/k-time signatures. Besides that, one-time/k-time signatures can be very large and introduce therefore much overhead.

5.1.2 Requirements

² Information dispersal algorithm (IDA) is a method to split information into n pieces in such a way that the information can be reconstructed from some predefined subsets of pieces.

All the proposed alternatives to conventionally sign each message in a multicast environment to provide data origin authentication are more or less good solutions regarding very specific requirements and features. None of the proposed alternatives is considered as best solution which provides non-reputable authentication for the general self-organizing network environment. Therefore, we will subsequently provide a list of requirements which must be satisfied to provide secure authentication for embedded P2P systems.

- arbitrary-to-arbitrary aware (broadcast, multicast, anycast, and unicast)
- loss-tolerant
- no additional latency (real-time aware)
- non-reputable
- fast and efficient
- scalable
- energy-aware
- mobility-aware
- resource efficient

Based on these requirements we will analyze existing, investigate promising and develop a suitable solution for data origin authentication in embedded P2P systems. This solution provides the foundation for the development of all other higher level security mechanisms and protocols required by this project to enable a secure and trustworthy embedded P2P middleware.

5.2 *Secure EP2P Networking*

The next step in providing a secure embedded P2P middleware is to achieve secure network composition, management and maintenance. Secure P2P network composition, management and maintenance assures that only legitimate, authorized and well-behaving peers are allowed to take part in a P2P network. The most essential part to provide this functionality is secure routing. Secure routing manages that only authentic, correct and efficient connections between the peers in the network are established. But without a suitable authentication mechanism, secure routing is not able to increase or provide security for an embedded P2P network. Now that we have defined the requirements for data origin authentication we are able to realize secure routing mechanisms based on an authentication mechanism satisfying these requirements.

Depending on the type of embedded P2P architecture, different variants of secure routing algorithms are possible in theory. The most natural architecture is the decentralized unstructured P2P network. On the other end of the scale the centralized structured P2P network can be found. In figure 4.1.2 the design space of P2P network architectures is illustrated. Also current representatives are shown.

The first realized type of P2P networks architectures was centralized and unstructured. These types have been used for file-sharing exclusively. The next type is decentralized and unstructured, often referred to as pure P2P network. The hybrid approach which is unstructured and combines aspects of centralized and decentralized networks is getting more attention. Also the DHT³ based approaches are becoming more attractive in the research community.

Each specific realization of P2P network architecture does have distinct requirements to secure routing. However, there are also some fundamental requirements which must be

³ Distributed Hash Table (DHT)

satisfied as otherwise it is not possible to establish a secure P2P network. Some of these requirements have already been identified by Rohatgi [33]. These requirements are listed subsequently.

- non-reputable peer authentication
- secure node ID assignment
- secure route determination
- secure routing table maintenance
- secure message forwarding
- intrinsic invariant verification
- privacy preserving

Additionally, some specific requirements emerge from the special application scenario of embedded P2P systems. This is because the target platforms of embedded P2P systems are heterogeneous by nature and vary from very resource constrained wireless sensor nodes with low bandwidth communication to computationally strong general purpose computers with broadband communication capabilities. A secure routing mechanism must also satisfy the following specific requirements to be useful:

- scalable
- resource efficient
- low bandwidth overhead
- fast path recovery
- energy aware
- multiple security levels
- group/role concept aware
- quality of service

Current secure P2P routing approaches [34], [35], [36], and [37] are only dealing with very specific problems – *DHT improvement, overlay architecture, ID provisioning* – and are not intended to be applied to embedded P2P systems. Therefore, the partners of this project will analyze existing secure P2P routing algorithms and protocols according to their applicability to embedded P2P systems. Based on the discovered requirements, possible solutions will be investigated and a suitable approach will be developed.

5.3 **Communication Security**

After a secure embedded P2P network has been created and the management and maintenance is assured, the possibility for secure communication between the peers and also between groups must be provided. . Communication security can be realized on two different layers – the network and the transport layer. Message authenticity, integrity, and confidentiality in conventional networks are usually achieved by means of end-to-end security mechanisms like IPSec [51], SSL/TLS [52] [53], or SSH [54]. Communication security in wireless sensor nodes cannot be achieved using the same technologies since the capabilities of wireless sensor nodes are very restricted. Hence, ongoing efforts towards a suitable solution for WSN communication security have already identified some possible solutions and provided prototype implementations [42], [43], and [44]. Recently also approaches towards link-layer security [39] and encryption have been investigated especially for wireless sensor networks [40], [41].

Providing communication security for embedded P2P systems seems to break down to two tasks, since the requirements for the security mechanisms are similar. The first is concerned with analyzing existing symmetric primitives regarding their computational and resource

requirements and also to investigate alternatives to common symmetric primitives for communication security in embedded P2P systems. In the second part the main challenge will be the design and implementation of feasible, efficient, and secure protocols for message encryption and transport. Such protocols must take into account the special constraints of embedded P2P systems, arising from the unprecedented amount of heterogeneity. They also must provide a mechanism to switch between different levels of security depending on the capabilities of the end nodes. Therefore, the following special requirements for communication security in embedded P2P systems must be satisfied:

- scalability
- different levels of security
- use of well known algorithms
- interoperability
- resource efficiency

5.4 *Application Level Protection*

Security requirements for EP2P-applications do not differ too much from conventional application-requirements. In principle, the same needs must be satisfied and applications should be kept as independent as possible from underlying technologies and architectures. The most important issue which has to be addressed in both traditional- and P2P-applications is access control. Access control is the central mechanism to control the way how users and system components communicate and interact with other systems and resources. Certainly, access control is also essential to protect systems and resources from unauthorized access. Many different architectures, models and implementations already exist [45], [46], [47], [48], [49], [50] emerging from this inherent application requirement. But beside access control, other relevant security issues have to be kept in mind :

- data integrity
- reliability
- availability
- accountability

The basic concept and functional structure of SMEPP is shown in D3.2 figure 4 [58]. In this figure, security is integrated in form of a cross cutting management-plane vertical to the layered structure of the SMEPP-middleware. The requested security level is passed by the application to the SMEPP-middleware by using simple API-calls which are directly forwarded to the vertical security manager to control all application-specific security-activities. However, the actual security related implementations are embedded into the SMEPP middleware or swapped out into protocols.

In contrast to conventional application-level security mechanisms, applications developed for EP2P middleware must implement an additional security feature. Applications must have the ability to specify and advertise their preferred- or available mechanisms to reach a specific security-level. This security level can be provided to and/or requested by the EP2P-application. Thereafter, the requested security level is evaluated by the security manager if it can be provided or an alternative security level has to be negotiated. A change in the security level is necessary if either the application or the middleware cannot guarantee the requested level of security. In a traditional point-2-multipoint P2P scenario, the security manager takes over the role of the security level negotiator. The security manager will evaluate the security level of the communication paths to the remote nodes and the available security levels of the remote applications itself. To evaluate the security level in a verifiable, coherent and correct manner, the middleware must also act as a communication- and security service provider for the P2P applications. Therefore, a P2P

application need not be aware of the actual communication mechanisms and communication partners as well as the underlying security mechanisms. The middleware will provide a security layer with stringent communication interfaces and a set of possible actions respectively commands for peer-to-peer applications. For the EP2P application it always seems that it is only communicating with the middleware.

A simple scenario based on the proposed approach is shown in Figure 5.4.1. There a peer (1) wants to use or provide a service from the EP2P system with security level high – in this simple example we don't specify the security level in more detail than low, medium and high, whereas in the actual implementation it will can be more detailed. The peer communicates with its middleware (2) which contacts the remote peers (3). The remote peers obtain the security levels from their application (4) to respont the required security level to the initiating middleware (5). The initiating middleware evaluates the possible security level and negotiates it with the middleware of the involved peers (6). Finally, the middleware provides the application with the result of the security level evaluation- and negotiation process. The application then has to decide if the available security level is adequate.

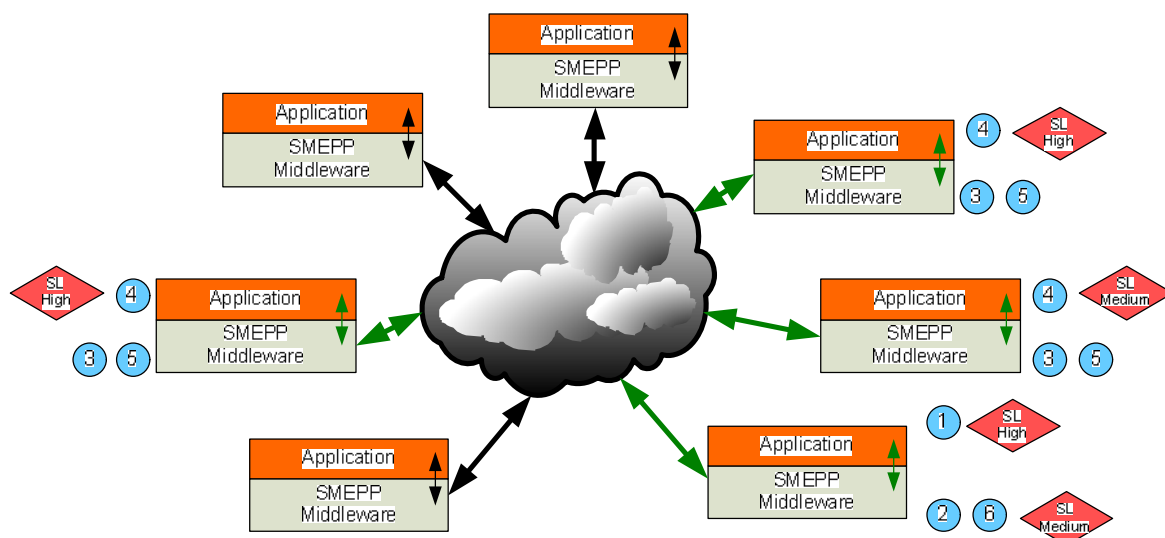


Figure 5.4.1: Application security level negotiation

Since all the actual communication and security mechanisms will be implemented in the middleware, it is possible to achieve a common set of security levels. These security levels must reflect comprehensive security requirements of typical applications. Therefore, a comprehensive list of security requirements must be generated. Thereafter, these requirements must be modeled in terms of security levels which can then be requested and provided by the applications.

Additionally, common preventive, reactive and passive perimeter security technologies – firewalls, intrusion detection and prevention systems, anti-virus software, anti-spam software – should be analyzed according their applicability to P2P systems and especially for embedded P2P systems. These technologies may provide some amount of security in addition to the previously mentioned mechanisms.

5.5 Security Management and Enforcement

Due to the nature of heterogeneous platforms it can be possible that P2P network functionalities cannot be performed on the same network layer. P2P routing – *for instance* – can either be implemented as an overlay network on top of IP, but can also be implemented natively at the ZigBee network layer (NWK) or as an overlay P2P object at the ZigBee application layer (APL). Let's assume that a set of such platforms can be part of the same heterogeneous network running the same level of routing services but based on platform-adapted mechanisms and maybe on different network layers.

A simple mechanism to overcome this problem is to run a management plane in parallel to the platform specific network services to be able to provide network management information to the right layer of the network stack by offering a common interface to the controlling middleware simultaneously.

Based on mechanisms of this management plane, it should be very easy to deploy pre-defined trust levels or to perform auditing, accounting or monitoring tasks. Required information can be gathered at appropriate locations and can be offered to the middleware in a unique style.

5.5.1 Security- and Trust levels

Trust levels represent the set of rights given to an external entity based on the system's knowledge about a specific entity. Entities of the same trust level are inherently grouped together into logical categories. For example, if a system's trust levels correspond to a specific user group, members of this group can only join this group by successfully closing an attestation process in advance to the usual authentication process. Attestation is necessary, since applications – *even knowing the right authentication secret* – can run in an un-trusted environment. Due to the manifoldness of possible platforms – *and their ability to support attestation processes* – different trust levels have to be defined and applied to entities to safeguard assets. An organization uses trust levels to define which external entities can interact with an entry point and which entities should have legitimate privileges to any given asset.

It is obvious that only trusted platform modules (TPMs) can be used to reach an adequate level of trust. But it is not expected that TPMs will be available on all SMEPP platforms. Therefore, alternatives will have to be designed, how singular TPM-enabled devices can be integrated into a heterogeneous network environment to form a trust anchor to assure an assessable level of trust in a specific user group.

Beside real TPMs, alternative solutions are needed to perform attestation mechanisms in non TPM-enabled environments.

While coping with trust is very challenging, security levels are much more straight-lined and easy to measure.

5.5.2 Specification, Decision and Delegation

As mentioned above, security can be measured very easily in terms of cryptographic algorithms or key length. But even if trust is closely related to security, we need additional mechanisms to define the requested level of trust – *as an anchor* – to have a reference point for security measurements.

Especially in heterogeneous environments, it is very important to declare the actual level of trust and security, since network devices can vary in computational performance. To overcome the problem of different performance and hardware support, a common understanding of trust- and security levels has to be defined. Based on these predefined levels, agreements can be negotiated to reach the same level of trust by using different mechanisms on different platforms.

Security-level- and trust-level matrixes have to be defined to enable a group-wide understanding of the achievable trust- and security standard. Due to the self-organizing nature of ad-hoc and sensor networks, a centralized decision point is not applicable. Thus, taking decisions, whether a new client is allowed to join an existing group has to be delegated to the group members or favored nodes of the group.

5.5.3 Deployment and Enforcement

As soon as network device attestation and authentication are completed, new routing information or new security services have to be deployed onto the right node or security element. Therefore, a framework is needed to establish secure channels between delegated management entities (decision points) and the deployment target (enforcement point). Only trusted nodes with the appropriate security level and authorization must be allowed to deploy security related information. The security related information which must be deployed can vary from very basic authentication and routing properties to complete security services which affect the behaviour of the complete embedded P2P system. Also the distribution of specific content to various nodes in order to introduce redundancy and increase the availability of this content as well as the reliability of the whole embedded P2P system can be seen as deployment of a service or security related information in a broader sense.

The enforcing of security related information and especially security policies must be an integral component of each embedded P2P node. The embedded P2P middleware must contain functionalities for enforcing security related information on different layers since the middleware manages the behaviour of the embedded P2P node as well as the whole system up from the data-link layer to the application layer. Therefore, it is necessary to start a process to identify the relevant intrinsic security features each middleware and therefore each node should provide. These security features must then be analyzed regarding their capability of being parametrized and managed by the system itself in a distributed manner.

5.5.4 Auditing and Monitoring

At least as important as deploying routing information, policies or rules is the monitoring of the enforced new properties. The deployed and enforced security services and security related information should be checked by the system after some time in order to detect misbehaving or adversary nodes in the embedded P2P system. Since it is not clear how to detect misbehaving and adversary nodes in embedded P2P systems as well as what to do with identified malicious nodes, the theoretic and practical foundations must be identified and specified first. Thereafter, such a functionality must be integrated into the embedded P2P system.

Each node in the embedded P2P system must be able to audit its intrinsic security capabilities and monitor the deployed and enforced security properties of the system itself. This can be achieved by either active traffic generation or passive traffic sniffing. Both techniques can be used to obtain relevant information which can thereafter be used in a

correlation process to identify possible problems in the system. Therefore, the relevant information which should be obtained through monitoring must be identified and an adequate correlation information exchange format, which allows reasoning on the content, must be specified.

5.6 Protection Against Side-Channel Analysis

Side-channel analysis has already been identified as a threat for EP2P systems. The most important attacks involve execution time (timing attacks), power consumption (power analysis), and electromagnetic emanations (EM analysis). As power consumption and electromagnetic emanations are closely linked, most countermeasures can also provide some protection against both.

While protection against timing attacks can be achieved relatively easily, e.g. with constant-time algorithms, the defense against power and EM analysis is much harder. The principal techniques for protection are hiding and masking [55]. Hiding countermeasures try to minimize the effect of executed operations and processed data values on the power consumption. Typical examples include randomized execution of operations and the use of dedicated, power-analysis resistant logic styles. Masking techniques on the other hand try to avoid the processing of critical data values altogether by splitting these values in two or more shares. Thereby, each share must be statistically independent from the original value, which can only be reconstructed by combining all of the shares. The basic idea is to split the input data of a cryptographic algorithm into shares and to process these shares independently. At the end the cryptographic output is reconstructed from the transformed shares. Masking techniques have been proposed for software implementations as well as hardware implementations (both at the architectural and the logic cell level) [55].

Experience has shown that a single type of countermeasure is normally not enough to deliver a satisfactable protection from side-channel analysis attacks. Therefore it is desirable to combine different types of countermeasures to increase the level of implementation security. The state-of-the-art for software implementations is appropriate masking and the randomization of these operations (through shuffling of operations as well as insertion of dummy operations). An example for a protected implementation on a low-cost 8-bit platform has been presented in [56]. The application of hardware countermeasures depends on the affordable design effort and cost as well as on device constraints like performance and energy/power consumption. They can range from the inclusion of noise generators in a device to the design a custom logic-cell library or full-custom design of the device (e.g. SABL [57]).

In EP2P systems, low-cost devices like sensor nodes will be restricted to a limited set of software countermeasures like the ones in [56]. More powerful devices might be custom-built for specific applications and can incorporate different forms of hardware countermeasures. The employed protocols and security policies need to be able to handle the compromise of the keys of several participating devices without immediate catastrophic breakdown of security measures.

5.7 References

- [1] Y. Challal, H. Bettahar, A. Bouabdallah, A taxonomy of multicast data origin authentication: issues and solutions, *IEEE Communications Surveys and Tutorials* 6 (3), 2004, pages 35-57.

- [2] Y. Desmedt, Y. Frankel, and M. Yung, "Multi-receiver/Multisender Network Security: Efficient Authenticated Multicast/Feedback", IEEE INFOCOM'92, 1992, pages 2045-54.
- [3] K. Kurosawa and S. Obana, "Characterization of (k, n) Multi-receiver Authentication," Information Security and Privacy, ACISP'97, LNCS, vol., no. 1270, 1997, pp. 204-15.
- [4] S. Obana and K. Kurosawa, "Bounds and Combinatorial Structure of (k, n) Multi-receiver A-codes," Designs, Codes and Cryptography, vol. 22, no. 1, 2001, pp. 47-63.
- [5] R. Safavi-Naini and H. Wang, "New Results on Multi-receiver Authentication Codes," Advances in Cryptology: EURO-CRYPT'98, LNCS vol., no. 1403, 1998, pp. 527-41.
- [6] R. Safavi-Naini and H. Wang, "Multireceiver Authentication Codes: Models, Bounds, Constructions, and Extensions," Information and Computation, vol. 151, 1999, pp. 148-72.
- [7] H. Fujii, W. Kachen, and K. Kurosawa, "Combinatorial Bounds and Design of Broadcast Authentication," IEICE Trans., E79-A vol., no. 4, 1996, pp. 502-06.
- [8] R. Canetti et al., "Multicast Security: A Taxonomy and Efficient Constructions," INFOCOM, 1999.
- [9] D. Boneh, G. Durfee, and M. Franklin, "Lower Bounds for Multicast Message Authentication," Eurocrypt '01, LNCS vol., no. 2045, 2001, pp. 437-52.
- [10] F. Bergadano, D. Cavagnino, and B. Crispo, "Individual Single-Source Authentication on the Mbone," IEEE Int'l. Conf. Multimedia and Expo, 2000.
- [11] A. Perrig et al., "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," IEEE Symp. Security and Privacy, 2000.
- [12] A. Perrig et al., "The TESLA Broadcast Authentication Protocol," RSA CryptoBytes, vol. 5, Summer 2002.
- [13] A. Perrig et al., "Efficient and Secure Source Authentication for Multicast," 8th Annual Internet Society Symp. Network and Distributed System Security, 2001.
- [14] A. Perrig et al., "SPINS: Security Protocols for Sensor Networks," Wireless Networks, vol. 8, 2002, pp. 521-34.
- [15] M. Mitzenmacher and A. Perrig, "Bounds and Improvements for BiBa Signature Schemes," Technical Report (TR-02- 02), Harvard University, 2002.
- [16] A. Perrig, "The BiBa One-time Signature and Broadcast Authentication Protocol," 8th ACM Conf. Comp. and Commun. Security, Nov. 2001.
- [17] L. Reyzin and N. Reyzin, "Better than BiBa: Short One-time Signatures with Fast Signing and Verifying," 7th Australian Conf. Info. Security and Privacy, LNCS vol., no. 2384, 2002, pp. 144-53.
- [18] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," Information and Computation, vol. 165, no. 1, Feb. 2001, pp. 100-16.
- [19] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," Advances in Cryptology, CRYPTO 97, 1997.
- [20] Y. Challal, H. Bettahar, and A. Bouabdallah, "A 2 Cast: An Adaptive Source Authentication Protocol for MultiCast Streams," IEEE-ISCC'2004, June 2004.
- [21] H. Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications," July 2003, RFC 3550.
- [22] Sara Miner and Jessica Staddon, "Graph-Based Authentication of Digital Streams," IEEE Symp. Security and Privacy, 2001.
- [23] P. Golle and N. Modadugu, "Authenticating Streamed Data in the Presence of Random Packet Loss," NDSS'01 : The Network and Distributed System Security Symp., 2001.
- [24] V. Paxson, "End-to-End Internet Packet Dynamics," IEEE/ACM Trans. Net., vol. 7, no. 3, June 1999, pp. 277-92.
- [25] M. Yajnik et al., "Measurement and Modeling of the Temporal Dependence in Packet Loss," INFOCOM'99, Mar. 1999, pp. 345-52.
- [26] C. K. Wong and S. S. Lam, "Digital Signatures for Flows and Multicasts," IEEE ICNP'98, Oct. 1998.
- [27] C. K. Wong and S. S. Lam, "Digital Signatures for Flows and Multicasts," IEEE/ACM Trans. Net., vol. 7, no. 4, Aug. 1999.

- [28] J. M. Park, E. K. P. Chong, and H. J. Siegel, "Efficient Multicast Packet Authentication Using Signature Amortization," *IEEE Symp. Security and Privacy*, 2002, pp. 227–40.
- [29] J. M. Park, E. K. P. Chong, and H. J. Siegel, "Efficient Multicast Stream Authentication Using Erasure Codes," *ACM Trans. Info. and Sys. Security*, vol. 6, no. 2, May 2003, pp. 258–85.
- [30] A. Pannetrat and R. Molva, "Authenticating Real-Time Packet Streams and Multicasts," *7th Int'l. Symp. Comp. and Commun., ISCC'02*, July 2002, pp. 490–95.
- [31] S. Even, O. Goldreich, and S. Micali, "On-line/Off-line Digital Signatures," *Advances in Cryptology – Crypto'89, LNCS vol., no. 435*, 1990, pp. 263–75.
- [32] S. Even, O. Goldreich, and S. Micali, "On-line/Off-line Digital Signatures," *J. Cryptology*, vol. 9, no. 1, 1996, pp. 35–67.
- [33] P. Rohatgi, "A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication," *6 th ACM Conf. Comp. and Commun. Security CCS'99*, Nov. 1999, pp. 93–100.
- [34] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Secure routing for structured peer-to-peer overlay networks. In *5th Symposium on Operating System Design and Implementation (OSDI 2002), December 9-11, 2002, Boston, Massachusetts, USA*.
- [35] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Proceedings of the First International Peer to Peer Systems Workshop (IPTPS)*, pages 261.269, Cambridge, MA, USA, Mar. 2002.
- [36] H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta. Limiting Sybil attacks in structured peer-to-peer networks. Technical Report NASTR-0017-2005, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, July 2005.
- [37] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending against Eclipse attacks on overlay networks. In *Proceedings of the 11th ACM SIGOPS European Workshop*, pages 115.120, Leuven, Belgium, Sept. 2004.
- [38] Marc Sánchez Artigas, Pedro García López, Antonio F. Gómez-Skarmeta. A Novel Methodology for Constructing Secure Multipath Overlays. *IEEE Internet Computing* 9(6). 2005, pp. 50-57.
- [39] Mohamed G. Gouda, E.N. Elnozahy, Chin-Tser Huang, and Tommy M. McGuire. Hop integrity in computer networks. *IEEE/ACM Transactions on Networking*, 10(3). pages 308-319, June 2002.
- [40] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*. November 2004.
- [41] Shiqun Li, Tiejian Li, Xinkai Wang. Efficient Link Layer Security Scheme for Wireless Sensor Networks. *Journal of Information and Computational Science*, Binary Information Press, 2007.
- [42] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. SPINS: Security Protocols for Sensor Networks. In *Proceedings of the 7th Ann. Int. Conf. on Mobile Computing and Networking*, pages 189–199. ACM Press, 2001.
- [43] S. Slijepcevic, V. Tsiatsis, S. Zimbeck, M. Srivastava, and M. Potkonjak. On communication security in wireless adhoc sensor networks. In *11th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 139–144, June 2002.
- [44] Yee Wei Law, Jeroen Doumen, Pieter Hartel. Benchmarking Block Ciphers for Wireless Sensor Networks. *MOSS*. October 2004.
- [45] David D. Clark, David R. Wilson. A Comparison of Commercial and Military Computer Security Policies. *IEEE Symposium on Security and Privacy*, 1987.
- [46] David E. Bell and Leonard J. LaPadula, *Secure Computer System: Unified Exposition and MULTICS Interpretation*, MTR-2997 Rev. 1, The MITRE Corporation, Bedford, MA 01730. March 1976.

- [47] David E. Bell and Leonard J. LaPadula. Secure Computer Systems: Mathematical Foundations, ESD-TR-73-278, Vol. I, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, MA 01731. November 1973.
- [48] Leonard J. LaPadula and David E. Bell. Secure Computer Systems: A Mathematical Model, ESD-TR-73-278, Vol. II, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, MA 01731. November 1973.
- [49] David E. Bell. Secure Computer Systems: A Refinement of the Mathematical Model, ESD-TR-73-278, Vol. III, Electronic Systems Division, Air Force Systems Command, Hanscom AFB, Bedford, MA 01731. April 1974.
- [50] Biba, K. J. Integrity Considerations for Secure Computer Systems, MTR-3153, Mitre Corporation, April 1977.
- [51] IPSec protocol family. Specified in Internet standard RFCs 2367, 2403, 2404, 2405, 2410, 2411, 2412, 2451, 2857, 3526, 3706, 3715, 3947, 3948, 4106, 4301, 4302, 4303, 4304, 4305, 4306, 4307, 4308, 4309, 4478, 4543, 4555, 4621, 4806, and 4809. Available online at <http://www.faqs.org/rfcs>.
- [52] Alan O. Freier, Philip Karlton, and Paul C. Kocher. Draft of SSL 3.0 Specification. Available online at <http://wp.netscape.com/eng/ssl3/>.
- [53] Tim Dierks and Eric Rescorla. The TLS Protocol, Version 1.2. Available online at <http://www.ietf.org/html.charters/tls-charter.html>.
- [54] Secure Shell (SSH). Specified in Internet standard RFCs 4250, 4251, 4252, 4253, 4254, 4255, 4256, 4335, 4344, 4345, 4419, 4432, 4462, 4716, and 4819. Available online at <http://www.faqs.org/rfcs>.
- [55] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power Analysis Attacks – Revealing the Secrets of Smart Cards. ADIS Book Series. Springer, 2007. ISBN 0-387-30857-1.
- [56] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In *Jianying Zhou, Moti Yung, and Feng Bao, editors, Applied Cryptography and Network Security, Second International Conference, ACNS 2006*, volume 3989 of Lecture Notes in Computer Science, pages 239–252. Springer, 2006.
- [57] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *28th European Solid-State Circuits Conference - ESSCIRC 2002, Florence, Italy, September 24-26, 2002, Proceedings*, pages 403–406. IEEE, September 2002.
- [58] Jarmo Kalaoja, Tuomas Paaso Jyri Toivonen. D3.2 Conceptual Architecture of Secure EP2P Middleware. IST Project n^o: FP6-IST-033563.

6 Using Cryptography to Avoid Threats in EP2P Systems

An Embedded Peer-to-Peer system faces many threats that can hinder the functionality of the whole system. Such threats may target the underlying infrastructure, such as highly constrained devices like sensor nodes, or attack the protocols and services that provide the EP2P services, such as group management services. It is then necessary to use security tools that can help into eliminating or mitigating the effects of such threats. Those tools are the security algorithms (primitives) and protocols, as seen in Section 4. Nevertheless, cryptographic primitives alone are only the foundation, but not the whole solution, to adequately protect an infrastructure. It is important to re-enumerate what are the threats that an EP2P system may face but from the point of view of some generic middleware and application requirements in a EP2P context, and to indicate the role of cryptography in the protection of those requirements against the possible threats.

Furthermore, due to the associated constraints of the embedded devices in an EP2P system, it is necessary to make a preliminary analysis of the load that such primitives impose over the whole network in terms of memory consumption and computational requirements. This is to ensure that the essential cryptographic primitives can be applied to these extremely constrained environments.

6.1 *Cryptography and Middleware/Application Threats in EP2P Systems*

On an EP2P system, the basic security requirement is the *protection of the information flow* against any external intruder. That is, when two devices that belong to the network communicate, the information that is exchanged should not be eavesdropped, that is, read by any entity that does not belong to the network (confidentiality) and should not be manipulated by non-authorized entities (integrity). Moreover, the source of the information should be known as a certain member of the system (authentication), avoiding low-level impersonation attacks.

The existing cryptographic primitives are able to fulfill such requirements easily. Confidentiality is assured by the existence of a symmetric encryption algorithm, given that the two entities that communicate with each other know a common security credential (i.e. secret key). Integrity is guaranteed by using hash functions, special SKC operation modes like CMAC, or mere checksum codes protected alongside with the data.

Finally, authentication could be easily achieved by using cryptographic protocols such as digital signatures: If a message is signed by the sender, the destination can use the public key contained in the certificate of the sender to test whether that message really comes from it. However, digital signatures are an expensive operation (cf. Section 6.2), and it may be problematic to sign every message if the network load is high. As a result, it can be possible to use digital signatures only for negotiating the security credentials between peers from time to time, and consider the existence of a correct security credential as a proof of the membership of a device.

Assuring the confidentiality, integrity, and authentication of the messages is able to stop threats such as message modification, message injection, node impersonation, and others. Still, there are other threats, such as message replaying, which cannot be stopped just by using these cryptographic primitives. Other methods, like the use of a session token, a security nonce or a message timestamp, have to be taken into account. Another problematic situation that may arise is to provide end-to-end protection for an information flow, not just

a mere hop-by-hop protection. It should be necessary for both origin and destination to negotiate a common secret key, mainly by using PKC, and use that resulting key to assure the confidentiality and integrity of the information.

Another important requirement is the existence of a mechanism to *control the access of any device* into the network. That is, only devices that belong to the network should be able to access its services. These devices should have an authorization (e.g. a certificate made by a higher authority) that identifies them as a part of the EP2P system, or a specific physical characteristic (like a RFID mechanism) that can identify the device. On the other hand, if a certain device is carried or controlled by a human user, such user should be the one that must be identified as a part of the network, e.g. by using biometrics, and should receive some credentials that identify him/her as a previously identified member of the system.

Once the communication channel has been protected, and the access issues managed, it is possible to establish the foundation of the EP2P system: The existence inside the system of groups and subgroups. Once a user enters a certain group, it should be able to publish its own services and/or to access the services offered within the group. Therefore, it is necessary to *manage how users join, publish, use and leave groups*. A suitable solution would be to consider the joining operation as the “entry point” of the system, in the sense that in the joining process the user will acquire the necessary credentials and permissions needed to belong to the group. Such credentials must allow the user to authenticate itself as part of the group, to access the communications of the group, and to change the state of the group itself, for example by publishing services. It should be noted that after leaving a group, the user should not be able to access it anymore.

Notice that closely related to the access control is the *authorization control*. Inside groups, a certain group may specify a policy in which only certain members that possess the adequate credentials can be able to be part of the group. Moreover, a certain user may specify a content filter, specifying that the members of a certain group may access all its information, whereas the members of another group can only access a subset of it. A possible solution that can be useful in this context, taking advantage of the existence of PKC, is the use of attribute certificates [1].

Not very related to cryptography, but important nonetheless, is the *auditing system*. Such a system helps to identify the actual state and behavior of the EP2P network and its members, e.g. providing the network topology or informing about the QoS. This is essential due to the changing nature of the network, where a member can appear, change its physical or logical location, and disappear, anytime, anywhere. The auditing system is in fact the basis of another monitoring subsystem, the *incident register*. It is highly recommended to have a group of methods and procedures that can store and inform about any strange behavior inside the EP2P system. The incident register can provide an invaluable help in detecting and locating any malfunctioning element, being the failure intentional or not.

Both the information produced by the auditing system and the incident register should be accessed only by the relevant users of the EP2P system, thus there is the need to manage inside the groups the security credentials needed to protect the confidentiality of this kind of information. Regarding the architecture itself of these systems, it is most probable that there is no need at all to develop a full Intrusion Detection System [2], although the existence of lightweight mechanisms and trust measurements for just detecting problems inside the system like DoS attacks or possible “node compromise” situations should be discussed.

Finally, as part of a dynamically reconfigurable network, it is needed to provide certain means to change the behavior of the elements of the network. This is a problematic

situation, where any external entity could easily affect the system injecting malicious or non-functional code, therefore opening a door to many “node compromise” attacks. For *securely updating the behavior of the network*, the use of digital signatures is a must. A problem that needs to be solved in the context of an EP2P network is who should be authorized to make such changes, and how the node can check whether a certain order comes from a trusted source.

There are many other aspects that make use of cryptography as a foundation for securing its internal behavior but need to be robust by themselves. This is the case of the *routing protocols*, that have to be able to withstand the existence of faulty nodes or connection problems while maintaining a certain QoS for important data, the *data aggregation processes*, that have to take into account the existence of faulty members reporting inconsistent readings, or the *service discovery and provisioning protocols*, that need to adequately provide all the existent services inside (or even outside) an EP2P group even if the situation of the network becomes unstable.

6.2 Suitability of Cryptography in Constrained EP2P Devices

A prerequisite for securing the protocols and services of the EP2P network is to have the security primitives (PKC, SKC, hash functions) included inside the devices in the form of hardware or software implementations. One of the features of this type of networks is the heterogeneity of their elements, that is, the different computational resources (such as CPU speed, memory, etc.) available to the devices. Sensor nodes, the main element of sensor networks, possess the lowest capabilities of all possible EP2P devices: The typical sensor node has an 8-bit microcontroller of 8Mhz, with 10Kb of RAM and 48-128Kb of instruction memory (ROM). Therefore, since it is necessary to protect the EP2P services, it is mandatory to check if these nodes are capable of timely executing the primitives.

In the case of *Symmetric-Key Ciphers (SKCs)*, the execution of the primitives should not provoke a penalty in communication. That is, the time dedicated to the execution of such software primitives must be under a period of time known as byte time, which represents the time required for sending a single byte of data, because on the contrary it could cause a delay in the sending of packets over the radio under a high throughput. For example, for a CC1000 transceiver with a bandwidth of 19.2Kbps, the byte time is approximately 420 μ s, and for a CC2420 (using the IEEE 802.15.4 standard) with a bandwidth of 250kps, the byte time is closer to 32 μ s.

Ganesan et al. [3] discovered in 2003 that the time required for encrypting a single plaintext was much less than the time needed for executing hash primitives in sensor nodes. Nonetheless, the average code size of a single primitive is less than 4000 bytes of ROM. For example, the encryption time of a plaintext with the stream cipher RC4 was 6 μ s, with RC5 was 26 μ s and with IDEA was 21 μ s. Note that for digesting 64 bytes with the SHA-1 algorithm it was necessary to wait 7777 μ s.

Later, in 2004 SKE primitives were introduced by means of a cryptographic package specifically created for early sensor node designs. This package, TinySec [4], provided the Skipjack primitive, which needed 48 μ s for encrypting a byte, and the RC5 primitive, which needed 33 μ s for encrypting a byte. This implementation also demonstrated the viability of using SKC to generate integrity fields (MAC).

Lastly, in 2006, Wei Law et. al. [5] and Jun Choi and Song [6] carried out a deep analysis on the encryption overhead of other instances of SKE primitives. Their studies

demonstrated that an optimized Skipjack improved in both encryption overhead (25 μ s) per byte and in memory overhead (2600 bytes of ROM) than the rest of algorithms including AES (an average of 8000 bytes). Nonetheless, RC4 achieved in terms of memory overhead 428 bytes, thus it was much better than an optimized Skipjack for extremely constrained devices.

On the other hand, the possibility of using *Public-Key Ciphers (PKCs)* in highly-constrained context was considered and even labeled, as “not possible”. However, Gura et. al.'s studies in [7] opened a door of possibilities for the applicability of PKC in sensor nodes. They demonstrated that Elliptic Curve Cryptography (ECC) was an appropriate technique for implementing PKC in highly-constrained context, since it offers a good functionality both in computation and memory storage, minimizing energetic costs and functional complexity. The cause of this reduction is mainly its small key size and its faster computation.

At present, there are several implementations which use certain ECC optimizations that allow their existence on sensor nodes. For instance, Liu and Ning implemented TinyECC [8], and Wang and Li implemented WMECC [9]. Concretely, TinyECC has a set of implementations in various elliptic curve domains (sec128r1, secp128r2, secp160k1, secp160r1, secp160r2, secp192k1, and secp192r1) according to the Standards for Efficient Cryptography Group (SECG) [10], while WMECC only has the implementation on the secp160r1 elliptic curve domain. Both implementations have a memory consumption of around 26Kb of instruction memory and 1.5Kb of RAM, generating a public key signature in about 1.3 seconds and verifying a signature in about 2 seconds.

As a conclusion, the execution of Symmetric-Key Ciphers in software should not pose a problem for highly-constrained devices like sensor nodes. In contrast, Public-Key Ciphers, although feasible, are still too slow for being used as part of the normal communication system. Moreover, there is a high penalty in both SKC and PKC in terms of memory consumption. Therefore, it is necessary to review what primitives should be more optimal for highly constrained devices, and the possibility and benefits of using hardware implementations. A deeper discussion on these subjects can be found in deliverable “D4.2. Security Services and Primitives for EP2P Systems”.

6.3 References

- [1] S. Farrell, R. Housley, *"An Internet Attribute Certificate Profile for Authorization"*, Request for Comments 3281, IETF PKIX Working Group, April 2002.
- [2] R. Bace. *"Intrusion Detection"*. MacMillan Technical Publishing, 2000.
- [3] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, M. Sichitiu. *"Analyzing and Modeling Encryption Overhead for Sensor Network Nodes"*. In Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA 2003), San Diego (USA), September 2003.
- [4] C. Karlof, N. Sastry, D. Wagner. *"TinySec: a link layer security architecture for wireless sensor networks"*. Proceedings of 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore (USA), November 2004.
- [5] Y. W. Law, J. Doumen, P. Hartel. *"Survey and Benchmark of Block Ciphers for Wireless Sensor Networks"*. ACM Transactions on Sensor Networks, vol. 2, no. 1, pp 65-93, February 2006.
- [6] K. Jun Choi, J.-I. Song. *"Investigation of Feasible Cryptographic Algorithms for Wireless Sensor Network"*. Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT 2006). Phoenix Park (Korea), February 2006.

- [7] N. Gura, A. Patel, A. Wander. “*Comparing elliptic curve cryptography and RSA on 8-bit CPUs*”. In Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), Cambridge (USA), August 2004.
- [8] A. Liu, P. Kampanakis, P. Ning. “*TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 0.3)*”. <http://discovery.csc.ncsu.edu/software/TinyECC/>, February 2007.
- [9] H. Wang, Q. Li. “*Efficient Implementation of Public Key Cryptosystems on MICAz and TelosB Motes*”. Technical Report WM-CS-2006-07, College of William & Mary, October 2006.
- [10] SECG - Standards for Efficient Cryptography Group. <http://www.secg.org/>.

7 Glossary

Acronym	Definition
AES	<i>Advanced Encryption Standard</i>
CCM	<i>Counter with CBC-MAC</i>
CBC	<i>Cipher Block Chaining</i>
CMAC	<i>Cipher-based MAC</i>
CTR	<i>Counter</i>
DES	<i>Data Encryption Standard</i>
DHT	<i>Distributed Hash Table</i>
DoS	<i>Denial of Service</i>
DWSN	<i>Distributed Wireless Sensor Network</i>
ECC	<i>Elliptic Curve Cryptography</i>
EP2P	<i>Embedded Peer-to-Peer System</i>
GCM	<i>Galois/Counter Mode</i>
HWSN	<i>Hierarchical Wireless Sensor Network</i>
IDA	<i>Information dispersal algorithm</i>
IP	<i>Internet Protocol</i>
ISO	<i>International Organization for Standardization</i>
JTAG	<i>Joint Test Action Group</i>
MAC	<i>Message Authentication Code</i>
NIST	<i>National Institute of Standards and Technology</i>
OTP	<i>One-Time Pad</i>
P2P	<i>Peer to Peer</i>
PGP	<i>Pretty Good Privacy</i>
PKC	<i>Public-Key Cryptography/Cipher</i>
PPP	<i>Point-to-Point Protocol</i>
QoS	<i>Quality of Service</i>
SABL	<i>Sense Amplifier Based Logic</i>
SECG	<i>Standards for Efficient Cryptography Group</i>
SKC	<i>Symmetric-Key Cipher</i>
SMEPP	<i>Secure Middleware for Embedded Peer-to-Peer systems</i>
S/MIME	<i>Secure / Multipurpose Internet Mail Extensions</i>
SSH	<i>Secure Shell</i>
SSL	<i>Secure Sockets Layer</i>
TEA	<i>Tiny Encryption Algorithm</i>
TLS	<i>Transport Layer Security</i>
TPM	<i>Tamper Proof Module</i>
TTP	<i>Third Trusted Party</i>
UMTS	<i>Universal Mobile Telecommunications System</i>
WLAN	<i>Wireless Local Area Network</i>
WPAN	<i>Wireless Personal Area Network</i>
WSN	<i>Wireless Sensor Network</i>