Master's thesis

Two years

The Department of Information and Communication systems

Computer Engineering MA

A Peer to Peer Security Protocol for the Internet of Things
Secure Communication for the SensibleThings Platform

Hao Zhang
Abstract

With the rapid development of the Internet connected technologies and applications, people are keen on embracing the convenience and practicalities they brings. As all kinds of technologies improve, the Internet of Things matures and is able to provide more advanced services to people, which connects a variety of devices, systems and applications beyond traditional machine-to-machine. However, it covers a variety of devices, protocols and applications, which makes it much more complex than a normal network. Nevertheless, equipped with appropriate security solutions, the Internet of Things is promising to bring us more conveniences and be widely applied in our daily life. And like the main application areas, wireless sensor networks with a frequent but short communication character, it requires an efficient and flexible protocol to protect the information. To protect the traffic of the Internet of Things is the focal point of this research work. Although many protocols for the Internet have been put forward, it is still not enough to meet the increasingly complex requirements from applications. Many of them are not efficient enough to adapt the device diversity and timely communication environment. This research work is trying to address this problem, by proposing a peer-to-peer security protocol to satisfy this varied environment. Secure communication is implemented on an open sourced platform for the Internet of Things. The philosophy of the platform it implemented on is also inherited to this protocol and the implementation. It avoids unnecessary handshakes between entities, which makes it more efficient in a wireless sensor network. Modularization and unit test are adapted in implementation to enhance the robust of the system. Its dynamic security level adjustment feature satisfies the realistic demand on one platform this protocol is implemented on. Finally, with a comparison test and an analysis using the BAN logic, the result shows that the proposed protocol is efficient to meet the specific goals and applicable for the platform.

Keywords: Security Protocol, Internet of Things, Peer to Peer, Cryptograph.
Acknowledgements

First of all, I would like to thank Professor Tingting Zhang. As my supervisor at Mid Sweden University, she gave me the opportunity to come here to study and work on this research project. In addition, she also gave me direction on this project work.

Secondly, I want to express my sincere thanks to Stefan Forsström and Kardeby Victor. They give me a lot of help on this thesis work with utmost patience, especially on the implementation for the SensibleThings platform and my thesis writing.

Finally, I would like to thank my family and my friends that supported me in my life and studies. And I want to express my great gratitude to all those who have supported me and helped me.
# Table of Contents

Abstract ................................................................................................................................. ii  
Acknowledgements .............................................................................................................. iii  
Table of Contents ................................................................................................................... iv  
Terminology .......................................................................................................................... vi  
Acronyms vi

## 1 Introduction .................................................................................................................... 8
1.1 Background and problem motivation ............................................................................. 9  
1.2 Overall aim ................................................................................................................... 10  
1.3 Concrete and verifiable goals ....................................................................................... 10  
1.4 Scope ........................................................................................................................... 11  
1.5 Outline ......................................................................................................................... 12  
1.6 Contributions ............................................................................................................... 12

## 2 Theory ............................................................................................................................ 13
2.1 Security Concepts ......................................................................................................... 13  
2.2 Security in Internet of Things (IoT) ............................................................................. 14  
2.3 Security in Peer-to-Peer Network ............................................................................... 15  
2.4 Cryptography Algorithms ........................................................................................... 16  
2.4.1 Rivest-Shamir-Adleman (RSA) ............................................................................... 16  
2.4.2 Advanced Encryption Standard (AES) ................................................................... 18  
2.4.3 Rivest Cipher 4 (RC4) ........................................................................................... 20  
2.5 Secure Socket Layer/ Transport Layer Security (SSL/TLS) ...................................... 21  
2.6 SensibleThings Platform ............................................................................................. 23  
2.7 Java Cryptograph Architecture ..................................................................................... 25  
2.8 Protocol Analysis ......................................................................................................... 27  
2.8.1 Analysis Procedures ............................................................................................... 28  
2.8.2 Basic Notation ....................................................................................................... 28  
2.8.3 Logical Postulates ................................................................................................. 29

## 3 Methodology ................................................................................................................... 32
3.1 Knowledge Accumulation .............................................................................................. 32  
3.2 Design Philosophy ........................................................................................................ 32  
3.3 Design Patterns ............................................................................................................ 33  
3.4 Test-driven development ............................................................................................... 34  
3.5 Protocol Analysis .......................................................................................................... 34
A Peer to Peer Security Protocol for the Internet of Things  
– Secure Communication for the SensibleThings Platform  

Table of Contents  

4 Design and Implementation .................................................. 35  
4.1 Security Protocol .............................................................. 35  
4.1.1 Principles ...................................................................... 35  
4.1.2 Protocol Overview .......................................................... 36  
4.1.3 Procedure I: Registration ............................................... 37  
4.1.4 Procedure II: Join & Communication ............................... 40  
4.2 Secure Communication in SensibleThings Platform ............... 43  
4.2.1 General Structure ............................................................ 43  
4.2.2 Secure Communication Implementation ........................... 46  
4.2.3 Security Manager ............................................................. 48  

5 Results .................................................................................. 50  
5.1 Test environment ................................................................. 50  
5.2 Performance ....................................................................... 50  
5.3 Comparison with SSL ........................................................... 52  
5.4 Security Protocol Analysis .................................................. 53  
5.5 Secure Communication Analysis ........................................... 56  

6 Conclusions ........................................................................... 58  
6.1 Future Work ........................................................................ 59  

7 References ............................................................................. 60  

Appendix A: Logical reasoning process for security protocol  
(Procedure I) with BAN logic..................................................... 63  

Simplified protocol .................................................................... 63  
Idealization of protocol ............................................................... 63  
Assumptions for protocol ........................................................... 63  
Reasoning process for protocol .................................................. 64
Terminology

Acronyms

AS  Authority Center
AN  Authority Node
P2P  Peer to Peer
IoT  Internet of Things
AES  Advanced Encryption Standard
API  Application Programming Interface
CBC  Cipher-Block Chaining
CFB  Cipher Feed Back
CSP  Cryptograph Services Provider
CTR  Counter
DoS  Denial of Service
DCXP  Distributed Context eXchange Protocol
DH  Diffie-Hellman
DHT  Distributed Hash Table
ECB  Electronic Code Book
JDK  Java Development Kit
LAN  Local Area Network
NIST  National Institute of Standards and Technology
OFB  Output Feed Back
PCBC  Propagating Cipher-Block Chaining
UCI  Universal Context Identifier
WEP  Wired Equivalent Privacy
RUDP  Reliable User Datagram Protocol
SSL  Secure Socket Layer
TDD  Test Driven Development
TCP  Transmission Control Protocol
Mathematical notation

\[ P | \equiv X \quad P \text{ believes } X \]

\[ P \bowtie X \quad P \text{ sees } X \]

\[ P | \sim X \quad P \text{ once said } X \text{ at some time} \]

\[ P | \Rightarrow X \quad P \text{ has jurisdiction over } X \]

\[ P \leftrightarrow^K Q \quad P \text{ and } Q \text{ share key } K \]

\[ k \rightarrow p \quad P \text{ has public key } K \]

\[ p \leftrightarrow^x Q \quad P \text{ and } A \text{ share secret } X \]

\[ \{X\}_K \quad \text{This denotes the formula } X \text{ encrypted under the key } K \]

\[ < X >_K \quad \text{This denotes } X \text{ combined with the formula } Y \]
1 Introduction

There is no doubt that our world has been becoming smarter than ever. As more smart devices come onto the market, like the Google glass, Sony SmartWatch, remote control lamps and countless other small devices 'hiding' in the every corner of our daily life, people are more likely to and more easily able to do what they want, to discover 'new land', to share with the world. Something that cannot be ignored is that these small elements (devices) are all connected to a web or the Internet. This connection makes world more vibrant. This can be regarded as a generalized Internet of Things [1], or Internet of Everything, in which the network connects people, data, processes and things [3].

According to Gartner, there will be 26 billion units installed in the Internet of Things by 2020, which excludes the smart mobile phones, tablets and laptops [2]. Nowadays many product designers dream up various methods to connect different kinds of intelligent devices with inherent connectivity. Though by 2014, IoT is still in its infancy. The great potentials brought by the IoT provide a huge space for different kinds of applications. According to the survey [1], there are mainly four main application domains: (1) Healthcare; (2) Smart environment; (3) Logistics and transportation; (4) Social and personal. Most of the applications are intended to improve the quality of our lives, at work, in the hospital, when jogging, even when sick. There is no doubt that it has a promising future and immeasurable potential, waiting for people to exploit it.

Nowadays, standardized ultra-low power devices with wireless technologies, including Wi-Fi, RFID, ZigBee, Bluetooth and Cellular, are still the main components of the IoT. As time passes, powerful embedded devices such as smart phones and tablets will occupy the great part of the IoT. The different devices not only bring kinds of applications, but also many problems, especially in terms of privacy and security issues.

It is easy to discover that how easy it is to physically attack these devices, as most of the time these components are unattended. The second problem is that many devices use wireless communication, which makes it easier to eavesdrop the messages. The final important issue that should be pointed out is the limited hardware resources and energy. These two disadvantages make it difficult to implement complex security schemes to protect the system and the information.
In 2013, mass surveillance programs were disclosed by Snowden shocked the world, a former CIA and NSA employee [4]. This not only leads to the great attention by governments, but also makes residents to take more care about their privacy and information safety. To protect the network and ensure the integrity of the messages is now an important and meaningful job. No matter how powerful the surveillance device, or whether we can defend our privacy, it is still better than nothing.

1.1 Background and problem motivation
Currently, many systems and platforms have decided to pursue better efficiency and to better satisfy application demand. Considering the security and privacy issues, the basic goal is to ensure the integrity and authentication of the messages, so that users or devices can trust what they receive and can make sure that the messages have not been modified.

To achieve the integrity goal, standard cryptographic techniques are essential and handy tools. Having a good selection of these cryptographic techniques is the basis to ensure the security of the system, as non-appropriate choices will sensitive information leaking out. The application areas should also be taken into consideration, as it is difficult to find one solution that can be adapted everywhere. Especially in some IoT applications, sensors are the main components and they are not able to implement complex security schemes. So the protocol should be flexible and feasible enough to adapt to different situations.

Additionally, in order to give consideration to the authentication of the system, it is better to have an overview of the network structure of the IoT system. In most cases, distributed network are preferable to centralized, which will have bottleneck problems and low scalability. This allows us to ensure that the authentication centre will not be a good idea for the authentication of the users. So the protocol should have the ability to identify other users in the system without the authentication center.

In addition, as the main domain for the IoT, wireless sensor networks are widely applied not only in the industrial, but also popular in weather monitoring, e-health and home life, etc. This kind of application scenario has its own properties that the traffic of each transmission is relatively small as one sensor only focuses on one kind of data, such as the position data from a GPS a sensor. Although each session is short,
sensors will frequently send the source data in a long time within certain time intervals. The normal protocol is hardly satisfied with these specific applications, especially in terms of efficiency.

Special attention is required in regards to the password and secret key management. It cannot be avoided to use password to manage the secret key, private keys certificates and so on. However, many disclosures are caused by inappropriate management of the password. From a system security point of view, the vulnerability of the system may lead to the same problems as well. Either way, this protocol should be able to better avoid this problem.

1.2 Overall aim

This project’s overall aim is to design a distributed or peer to peer security protocol to protect the traffic of the IoT and implement it on the SensibleThings platform as secure communication. This thesis work aims to achieve the following goals:

- Provide a functioning laboratory prototype of the security protocol for the IoT, including all the necessary modules with basic functions.
- The security protocol is well evaluated through logic analysis
- Implemented secure communication is able to support different kinds of security levels for different security requirements.
- Standard interfaces for secure communication are provided for application developers to use.

Therefore the problem I will solve in this thesis is: designing a peer-to-peer security protocol to protect the traffic in the IoT, and implemented it on SensibleThings platform as an efficient means of communication.

1.3 Concrete and verifiable goals

To be able to achieve the above aims, a series of concrete and verifiable goals are needed for this thesis work. These goals show, step by step, the path to the final outcome, which could be seen as below.

(1) Do research on the Internet of Things and its security issues, peer to peer systems and cryptography. Summarize the concept and implementation of related technologies. Then find out the result of
former researches. Analyze the disadvantages and advantages of all possible solutions. Decide what has been done and what needs to be accomplished.

(2) Do research on former security protocols. Find out about the existing security protocol principles and vulnerabilities with targeted attacks. Consider possible modifications and improvements for this project. After that, have a solid understanding of the design philosophy of the SensibleThings platform and its practical problems. Next, consider the required features of this protocol. These features are: scalable, lightweight, fast, flexible, integrity, authenticatable and available. When requirements are clear enough, it is time to develop a prototype of a security protocol.

(3) Learn about the SensibleThings platform, a context aware distributed system developed for the IoT. Study the principle and architecture of the system, and implement the practices with the source code. Figure out each layer of the system, and decide how to implement the security protocol in this platform.

(4) Design and implement secure communication on the Sensible Things platform. The first and most important thing is to follow the principle of the platform. The less modification of the original system, the better. Each component should be modularized and less coupled. For the sake of stability and robustness, basic cryptographic functions and key management for secure communication should be fully tested.

(5) Test and evaluate secure communication with demo clients. Analyze the security protocol using BAN logic. The secure communication should also be analyzed from a practical viewpoint. Find out the potential risks and limitations of secure communication.

1.4 Scope

The research work concentrates on the design of a security protocol for the IoT, and the implementation of a corresponding secure communication on the SensibleThings platform. This protocol will not only cover the integrity of messages, but also the authentication of each client by providing an efficient authentication mechanism. In
order to satisfy the one transmission pattern that exist in wireless sensor networks, in which each transmission is short but frequent, one security level dynamic adjustment will be focused on. Finally, a logic analysis of this protocol is also included in this thesis work, as well as a functioning laboratory prototype of the security protocol for the IoT, including all the necessary modules and basic functions.

The security of a system actually includes lots of things. This proposed security protocol could be seen as the main part of it, however, not the whole part of it. As the time is limited, other security issues will not be covered in this thesis work, such as password management, file privilege management and security attack test against this protocol.

1.5 Outline
The structure of this thesis is organized as follows. Chapter 1 introduces the background and motivation of this research work, both overall goals and concrete goals. Chapter 2 introduces related work on the topics covered in this thesis. Chapter 3 introduces the chosen methodology and methods to accomplish the research work. Chapter 4 begins with a brief description the general structure of the security protocol. Then it describes the design and implementation of secure communication in detail. Chapter 5 gives a detailed evaluation of the system’s performance and carries out a logic analysis of this protocol. Chapter 6 summarizes the whole research work and points out future directions.

1.6 Contributions
The design of this security protocol is drafted by the author, is revised together with the supervisor. The SensibleThings platform for the IoT developed by Stefan Forsström, Victor Kardeby and their colleagues, provides the implementation space for this security protocol. Based on this platform, secure communication is developed, evaluated and analyzed by the author.
2 Theory

In this chapter, a short description of security concepts, security situation in IoT and P2P, cryptographic algorithms and the SensibleThings platform and BAN logic is given and discussed.

2.1 Security Concepts

NIST Computer Security Handbook defines the term computer security as follows:

The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications). [11]

This definition gives three key objectives which are the heart of computer security: Confidentiality, Integrity and Availability.

Normally, these three concepts are referred to as the CIA triad (Figure 1).

FIPS (Federal Information Processing Standards) 199 [13], give a more detailed description of the three terms, listed below.

Confidentiality: Information access and disclosure are restricted. The proprietary information and personal privacy are protected.

Integrity: Protecting the information from improper modification and destruction.

Availability: Ensure reliable and timely use and access to the information.
Although the CIA triad gives a clear definition of the security objectives, some experts think that additional concepts are needed[12]. Two of the most commonly mentioned are authenticity and accountability. Authenticity implies being able to be verified and trusted by others. So that others can make sure who they are talking to. Accountability indicates that actions in a system could be traced back only because of a later recovery and fault isolation requirements under certain protection.

### 2.2 Security in the Internet of Things (IoT)

Luigi Atzori et al.[1] conducted a thorough survey on the Internet of Things by addressing almost every aspect of the Internet of Things including different visions of the paradigm. It also gives a description related to the security and privacy issue. In regards to security, authentication and data integrity there two major problems. It is difficult to ensure authentication especially against active attacks, because it normally requires appropriate infrastructures and an authority center, working on exchanging appropriate messages with nodes. This is not available in some IoT scenarios. To ensure the data integrity, traditional cryptographic methodologies are preferred. While a problem occurs when come to the devices with limited resources in terms of bandwidth, energy and computation capabilities, which makes many cryptographic algorithms impossible. However, a few solutions are proposed with some light symmetric key cryptographic schemes.

Gan Gang et al. [5] address the existing security risks of the Internet of Things in terms of the network and transmission, and propose suggestions to solve these problems. They suggest to do the encryption in the gateway node to avoid the single abnormal node to implicate the whole network. Setting up of the sensor networks for IoT applications, which needs to connect external network should be done with care, because this is what brings the threat from external attacks, like the Denial of Service (DoS) attack. To achieve a high-level security, end-to-end encryption is preferred to by-hop encryption. This is because it exposes the messages’ source and destination, which is easy attacked by malicious access caused by analysis of the communication services.

Suo Hui et al. [6] reviewed the security in the Internet of Things. They inspected the security from secure architecture with four key levels: perceptual layer (also known as recognition layer), network layer, support layer and application layer. The perceptual layer mainly consists of sensors that collect all kinds of information. This layer is weak, as it is difficult to set up security protection systems on limited hard
ware resources. On the second level – the network layer-information is transmitted on few basic networks. This level is very important to the IoT, which should be given complete safety protection against attacks like man-in-the-middle attack and counterfeit attack and so on. The support layer will set up reliable platforms for applications. Due to the mass data that will be processed in this layer, it is a challenge to recognize the malicious information. The application layer provides all kinds of services to users. As the requirements change, problems like data privacy, disclosure of information and access control should be handled with care.

Base on the MediaSense [8] framework, Sijan Kafle [9] proposed a security architecture or mechanism which is adaptive to distributed systems. This thesis work developed a proof of concept application and prototypes to verify their approach. The author also implemented security features from the security architecture as an extension to the MediaSense framework.

2.3 Security in Peer-to-Peer Network

The peer-to-peer (P2P) network has a very different networking concept to client-server relationships in terms of information exchange. Sandeep Kumar et al. [7] collected various attacks on P2P networks and given them a classification. When it comes to unstructured P2P systems, self-replication and man-in-the-middle attacks only should be cared about. When it comes to structured P2P systems, routing attacks, storage and retrieval attacks and node joins and leaves are common ways of attack. This paper also describes the various security issues of defense against general and specific attacks.

To enhance the security of P2P computing, Wang Min [10] presented a P2P security plan based on SPKI/SDSI (Simple Public Key Infrastructure/Simple Distributed Security Infrastructure) certificate in JXTA – one open source P2P distributed computing platform. Although it is simple for all distributed nodes to carry out, there is the limitation of scalability and bottleneck, as the need of the authentication center.
2.4 Cryptography Algorithms

As the cryptography algorithms always hold the fundamental basis of every security system, it is very important to pick an appropriate algorithm for each situation. With the development of the technology, the computation ability has been greatly improved. This makes some attacks possible to avoid by using old cryptography algorithms, like DES. In this section, some cryptography algorithms are briefly introduced, and each of them will be used in this thesis project.

2.4.1 Rivest-Shamir-Adleman (RSA)

The public-key cryptography is perhaps the true revolution. It is also known as asymmetric encryption. Unlike symmetric encryption one paired keys are used to encrypt and decrypt the messages and their corresponding algorithms. In theory, public-key algorithms are based on mathematical functions rather than on permutation and substitution. In broad terms, public-key crypto systems can be applied into three areas:

- Encryption/decryption: The sender encrypts a message with the receiver’s public key.
- Digital signature: The sender signs a message with a private key.
- Key exchange: Two ends communicate to exchange a session key.

The most widely used public-key crypto system is Rivest-Shamir-Adleman (RSA) cryptographic algorithm [12]. This is motivated by the published works of Diffie and Hellman many years earlier, who introduced a new approach to cryptography, but never truly developed it.
The difficulty of attacking RSA is based on the difficulty of finding the prime factors of a composite number[26]. The RSA algorithm makes use of an expression with exponentials. Figure 2 shows a summary of RSA algorithm. In this figure, Bob is going to encrypt the message using a RSA algorithm, and then send it to Alice. Before transmission, Alice generated a key pair including a private key and a public key. As shown in the figure, the private key consists of \{d, n\}, and the public key consists of \{e, n\}. Alice has published her public key, so Bob knows and can use this public key to encrypt the message \(M\). Bob calculates \(C = M^e \mod n\) then transmits \(C\). When receiving this cipher text, Alice decrypts by calculating \(M = C^d \mod n\). Then Alice knows exactly what Bob would like to tell her.

![Diagram of key generation, encryption, and decryption](attachment:image.png)

**Figure 2: The RSA Algorithm [12]**

The advantages of the RSA algorithm are:

- The primary advantage is providing security and convenience by solving the problem of distributing the key for encryption.

- It provides the ability of message authentication by digital signature. It would be impossible to modify a signed message without invalidating the signature.
The disadvantages of the RSA algorithm are:

- Important when using the asymmetric encryption is being able to trust the public key belonging to a certain entity. It is difficult to make absolutely sure. It is essential to verify that these public keys belong to a certain entity.

- Compared to symmetric encryption, the speed of encryption and decryption is relatively slow. It will require more computer resources.

- The loss of the private key may not only lead to serious security threat, but all cipher text encrypted with the corresponding public key cannot be decrypted.

2.4.2 Advanced Encryption Standard (AES)

Intended to replace DES as the approved standard, the Advanced Encryption Standard (AES) [12] was published in 2001 by the National Institute of Standards and Technology (NIST) [28]. AES is a block cipher, which uses a 128-bit block size and key size of 128, 192, or 256 bits. In the foreseeable future, a key size of 128 bits is considered secure [27]. Instead of using a Feistel structure, each full round of AES consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field, and XOR with a key.

Compared to public-key algorithms such as RSA, most symmetric ciphers including AES are quite complex. The plaintext size entered into the encryption and decryption algorithm is a 128-bit block. Figure 3 shows the AES encryption and decryption process in detail.

Unlike the Feistel structure, in which half of the data block is used to modify the other half of the data block and then the halves are swapped. The AES processes the whole data block as a single matrix in each round using substitutions and permutation. During the encryption and decryption, four different stages are used, one for permutation and three for substitution.

The first stages are substitute bytes using an S-box to do a byte-by-byte substitution. The second stage is to shift rows, which is a simple permutation. Then mix the columns, which makes use of arithmetic over
$GF(2^8)$. The final stage is to add the round key that the current block has a bitwise XOR with a part of the expanded key.

![AES Encryption and Decryption](image)

**Figure 3: AES Encryption and Decryption [12]**

The advantages of the AES algorithm:

- Compared to asymmetric algorithms, AES as a symmetric algorithm is simple and fast. Therefore it requires less computer resources.

- Unlike asymmetric algorithms, one key pair is used only for communication with one party than many. This feature isolates the security of different communication. If one key is compromised, communication with other people is still secure.

The disadvantages of the AES algorithms:
2.4.3 Rivest Cipher 4 (RC4)

As a widely used symmetric encryption, Rivest Cipher 4 (RC4) is a prominent stream cipher designed by Ron Rivest for RSA Security [12]. RC4 is also used in popular protocols such as Secure Sockets Layer (SSL)/Transport Layer Security (TLS) and Wired equivalent Privacy (WEP).

The RC4 algorithm is simple and could be easily explained. Figure 4 shows the logic of RC4. A 256-byte state vector $S$ is initialized by a variable-length key of from 1 to 256 bytes. Shown in the figure, these elements are $S[0]$, $S[1]$, …, $S[255]$. And $S$ includes a permutation of 8-bit numbers from 0 to 255. When encrypting and decrypting, a byte $K$ is generated from $S$. The entries in $S$ are permuted again after each value of $K$ is generated.

At the beginning, $S$ is equal to the values 0 to 255 in ascending order, which is, $S[0]=0$, $S[1]=1$, …, $S[255]=255$. At the same time, a temporary vector $T$ is generated. If the length of the key $K$ is 256 bytes, then $T$ is directly filled up by the key. Otherwise, $T$ will be filled up by the $T$ repeated as many times as necessary. After that, $T$ is used for creating the initial permutation of $S$. As soon as the $S$ vector is initialized, the input key will not be used any more. The stream generation involves cycling through all the elements in $S$. This process continues from $S[0]$ to $S[255]$. When encrypted, the next byte of the plaintext is XOR with the value $K$. In decryption, the next byte of cipher text is XOR with the value $K$. 

• Before using the secret keys, it should be guaranteed that these keys are exchanged securely.

• Each generated key can only be used for one party. This causes a problem in managing all the secret keys by ensuring their security.

• It cannot be verified which message has come from which party, due to both sender and receiver using the same key.

• It is susceptible to noise in transmission if using “feedback” modes. This means that, if one part of the data is messed up, is not recoverable.
The advantages of RC4:

- As a stream cipher, RC4 is much faster than a block cipher. And it requires less resource, since it works on only a few bits at a time.
- It does not need padding, which means problems caused by padding could not be avoided.

The disadvantages of RC4:

- Integrity protection and authentication are not guaranteed in this cipher.
- It could be susceptible to security problems if not implemented well, such as Wired Equivalent Privacy (WEP).

### 2.5 Secure Socket Layer/Transport Layer Security (SSL/TLS)

Secure Socket Layer is a layered protocol, which provides the security services between Transmission Control Protocol (TCP) and applications that use TCP [12]. The purpose of SSL is to make sure that TCP provides a reliable end-to-end secure service. It also provides privacy and reliability between two communicating applications [22]. Document RFC 6101 [12] gives complete description of this protocol. And the position of the protocol stack in the TCP/IP model could be seen from Figure 5. There are
two layers in this protocol. The lower level is layered on top of some reliable transport protocol, such as TCP. This level is called an SSL record protocol, which could be seen from Figure 5. Above this, there is an SSL handshake protocol, which allows the client and server to authenticate each other. And then these two will negotiate the encryption algorithm and cryptographic keys before their applications communicating between each other. As an independent application protocol, SSL protocol guarantees the transparency of the higher-level protocol based on it.

At the application layer, the ongoing messages contain fields for description, length, and content. These messages then will be taken over by SSL. Firstly, the data will be fragmented into manageable blocks. The data may also be compressed, which is an optional choice. Then a MAC is applied and then is encrypted. Finally transmitted to the destination. Receiver firstly will decrypt the data, then verify, decompress, and reassemble those blocks. After that, it will be delivered to the clients in higher level [22].

Same as the SSL protocol, the primary goal of Transport Layer Security protocol (TLS) is also to provide data integrity and privacy between two applications. Document RFC 5246 [23] gives a detail illustration of this protocol. TLS protocol also has two layers. One is the TLS record protocol, and another one is the TLS handshake protocol. The TLS handshake protocol is a bit like the SSL handshake protocol, which allows the client and server to authenticate each other and then negotiate the encryption algorithm and cryptographic keys before their applications communicating between each other. And protocol is the TLS record protocol that provides the connection security. It guarantees the reliability and privacy [23].
Figure 5: SSL Protocol Stack

2.6 SensibleThings Platform

The SensibleThings platform [14] is an open source platform for creating efficient and fast Internet Things applications. To become a common platform for communication between sensors and actuators on a global scale, and enable a widespread proliferation of IoT services, the creators of this platform addressed many of the practical difficulties it has to face. The requirements of the applications on the IoT from an underlying platform are described as below.

- Quickly disseminate the information from source to end points.

- Stable and well handled devices joining and leaving the system with high churn rates. No central point failure, it may be able to heal itself.

- Lightweight enough to run on devices with limited hardware resources, as in IoT there may be different kinds of devices with varying computational, bandwidth and storage resources.

- Adaptive and extensive enough to accept a wide range of possible devices and applications.

- Free and easy to adapt to the usage in commercial products.
The SensibleThings platform was created in order to address the stated requirements and solve the above problems, which is an implementation and realization of the MediaSense architecture explained in [8]. The architecture of the SensibleThings platform can be seen from Figure 6. It describes the different layers and components of the platform.

![Figure 6: Overview of the SensibleThings platform’s architecture [14]](image)

The layers designed in this platform include interface layer, add-in layer, dissemination layer, networking layer and sensor/actuator layer. Each layer is described briefly below.

- **Interface Layer**: This layer provides the generic Application Programming Interface (API) for applications that developers want to build on top of the platform. Through the interfaces layer, applications for IoT can interact with the SensibleThings platform.

- **Add-in Layer**: This layer provides the extensibility for developers to add optional functionality and optimization algorithms to the platform to meet specific application requirements. This pluggable add-ins can be dynamically loaded and unloaded when needed.

- **Dissemination Layer**: The dissemination layer is the fundamental layer in the SensibleThings platform, which enables all joined entities communicate with each other. This layer also implements a variant protocol based on the Distributed Context eXchange Pro-
protocol (DCXP). It provides communication among connecting entities by using the Universal Context Identifier (UCI). This layer is mainly made by three components: the dissemination core offering the primitive functions of the DCXP, the lookup service which resolves and stores UCIs among the system, and the communication component based on kinds of different communication protocols, like TCP, RUDP, SSL et al.

- Networking Layer: This layer provides the ability of entities to join to the platform under the Internet Protocol (IP) based infrastructure, such as wireless and mobile networks or the wired networks.

- Sensor and Actuator Layer: This layer offers the ability for sensors and actuators to connect to the platform in different ways. These sensors and actuators can be directly connected if they are accessible from the application. In addition, the sensor and actuator can connect based on the sensor and actuator abstraction component. So, this layers can be separated into five components: a direct sensor and actuator access component, a sensor and actuator abstraction component, different sensor and actuator networks, sensor and actuator gateways, and the physical sensors and actuators.

2.7 Java Cryptography Architecture

The Java platform, which was used for this project, strongly focuses on security, including many aspects like cryptography, language safety, secure communication, and access control.

The Java cryptography architecture [15] is a major work of this platform, which is developed based on a “provider” architecture and a series of APIs for various purposes, such as encryption (asymmetric/symmetric, block/stream ciphers), key generation and management, certificates and certificate validation and so on. These APIs provides easy ways for developers to integrate these security mechanisms into their applications. This application was designed around following three principles.

- Implementation independence: Application developers do not need to implement security algorithms. Security services are implemented in providers, which can be pluggable into the Java platform through the standard interfaces.
• Implementation interoperability: Security providers are interoperable with applications.

• Algorithm extensibility: The Java platform supports the custom providers that implement the series of security services.

To understand the architecture of the Java cryptography architecture, Cryptographic Services Provider (CSP) is a basic and important concept that should first be presented. Each CSP contains the provider's name and a list of the security algorithm/services it implements. When an application needs a particular algorithm, the JCA will search the provider's database, and if a match is found, the corresponding instance is created.

An example of a detailed specification is when an application needs to generate a digest of a message. It will then use the MessageDigest object in JCA, and calls a particular service (for example, the “MD5” algorithm). Then it will get the implementation from the one of the installed providers. The codes are shown below.

```java
md = MessageDigest.getInstance("MD5");

md = MessageDigest.getInstance("MD5", "ProviderC");
```

Two corresponding figures show the work flow of this process.(Figure 7). The figure presents three different providers that implement various message digest algorithms (“SHA-1”, “SHA-256”, “MD5”, and “SHA-512”). The providers are listed from left to right according to priority. The application calls for the “MD5” service. In the left figure, the application requests an MD5 digest algorithm implementation and it has not specified a provider name. So it will search the providers in order of preference order. We can see from the left figure where the first provider match is ProviderB. In the right illustration, the application calls the MD5 algorithm by specifying a provider's name. So the implementation from ProviderC is returned, although ProviderB has a higher preference order.
In order to use a specific algorithm implementation and also to avoid the poor support by the default providers, the Bouncy Castle Crypto APIs [16] are used in this thesis project. It provides the provider with the Java cryptographic architecture and the Java cryptography Extension.

2.8 Protocol Analysis

Burrows-Abadi-Needham logic, which is also known as BAN logic is a collections of rules and postulates for modelling and analysing the information security protocol [24]. This logic is also an logic of beliefs, since it analyses the authentication protocols by deriving the beliefs from a series of logical deductions [29]. It helps users analyse and discover whether the protocol designed is secure against eavesdropping, or trustworthy, or both.

BAN logic is able to help us answer the following questions [24]:

- What the aim of this protocol or what does this protocol to achieve?
- Are more assumptions needed for this protocol than for another?
• Is there any unnecessary part that could be left out without weakening this protocol?

• Is there any encrypted information that could be sent in clear text without weakening it?

BAN logic starts with the assumptions about the initial state of the protocol. A typical BAN logic could be used to verify the message origin, message freshness and the origin’s trustworthiness [25]. Postulates and definitions are used in the BAN logic analysis. A series of security protocol notation formulations are used for the protocol analysis, in which these formulations are adapted from the original paper.

2.8.1 Analysis Procedures

From a practical point of view, there are four main procedures for the analysis this logic. They are listed as follows:

• Idealize the protocol from the original one.

• Set the assumptions about the initial state of proposed protocol.

• Logical formulas are added to the each statement of the protocol

• Logical postulates are applied to the assumptions and assertions. And the purpose of it is to discover beliefs in this protocol that the parties had.

These procedures may be repeated when the idealized protocol is refined or new assumptions are found as necessary. Step by step, this logic points the direction of protocol analysis from the initial assumptions to the final beliefs.

2.8.2 Basic Notation

There are several sorts of objects: principals, encryption keys, and formulas (statements).

Typically, Symbols like A, B, and S denote specific pals; $K_{ab}, K_{bs}$ and $K_{bs}$ represent specific shared keys; $K_a, K_b$ and $K_s$ denote specific public keys, and $K_a^{-1}, K_b^{-1}$ and $K_s^{-1}$ are the corresponding private keys; $N_a, N_b$ and $N_c$ indicate specific statements. The symbols X and Y range over statements; P, Q and R range over principals; K ranges over encryption keys. The comma is used as conjunction for propositional connectives. In addition to conjunction, other construction are shown below:
• \( P| \equiv X \): P believes X, or P is entitled to act as though X is true.

• \( P \sim X \): P sees X. Someone has sent a message containing X so that he can read

• \( P| \sim X \): P once said X at some time, e.g. a used key K. Or A uttered a message containing X.

• \( P| \Rightarrow X \): P has jurisdiction over X. P maybe an authority over X and can be trusted on X.

• fresh( X ): X is fresh. It indicates that X has not been sent at any time before.

• \( P \leftarrow^K \rightarrow Q \): P and Q share key K. P and Q can use key K to communicate. The key is unknown to anyone else.

• \( K \rightarrowdot P \): P has public key K. P has a public key K, and the corresponding private key is \( K^{-1} \)

• \( p \leftarrow^{x} \rightarrow Q \): P and A share secret X. X is a secret only known by P, Q, or principals trusted by them. P and Q may use X to prove their identities to one another, e.g. X is a password for them.

• \( \{X\}_K \): This denotes the formula X encrypted under the key K. Formally, \( \{X\}_K \) is short for \( \{X\}_K \) from P.

• \( <X>_K \): This devotes X combined with the formula Y. Y is intended to be secret and its presence proves the identity of whoever utters \( <X>_K \). In implementations, X can simply be concatenated using the password Y. Y plays a special role as a proof of origin for X.

### 2.8.3 Logical Postulates

Here, the notation in the BAN papers [24], \( \frac{X}{Y} \) means if X is true, then Y is true. It assumes that participants in protocols are good logicians: if P believes a proposition X and \( \frac{X}{Y} \), then she believes Y too. This is true for the axioms also.

The *Message meaning rules* concern the interpretation of messages. They all explain how to derive beliefs about the original messages in the protocol.

For shared keys, it postulates
A Peer to Peer Security Protocol for the Internet of Things – Secure Communication for the SensibleThings Platform

That is, if P believes that the key K is shared with Q and sees X encrypted with the key K, then P believes that Q once said X.

For public keys, it postulates

\[
\frac{P \equiv Q \rightarrow P, P \equiv (X)_{K-1}}{P \equiv Q \rightarrow X}
\]

That is, if P believes that K is Q's public key, and P receives a message encoded with Q's secret key, then P believes Q once said X.

For shared secrets, it postulates

\[
\frac{P \equiv Q \equiv Y \rightarrow P, P \equiv <X>_Y}{P \equiv Q \rightarrow X}
\]

That is, if P considers that the secret Y is shared with Q and sees \(<X>_Y\), then P believe that Q has said X.

The *nonce-verification rule* tells the check whether a message is recent or not, and hence, if the sender still believes in it:

\[
\frac{P \equiv \text{fresh}(X)P \equiv Q \rightarrow X}{P \equiv Q \rightarrow X}
\]

That is, if P believes that X could have been told only recently and Q once said X, then P believes that Q believes X. For simplicity, X could be “cleartext”.

The *jurisdiction rule* expresses that if P believes that Q has jurisdiction over X then P accepts Q as true based on the truth of X:

\[
\frac{P \equiv Q \Rightarrow X, P \equiv Q \equiv X}{P \equiv X}
\]

If a principal sees a formula, then he also has the ability to see its components, provided he knows the necessary keys:

\[
P \equiv (X, Y)
\]

(6)

\[
P \equiv <X>_Y
\]

(7)

\[
P \equiv Q \rightarrow K \rightarrow P, P \equiv (X)_{K-1}
\]

(8)

\[
P \equiv K \rightarrow P, P \equiv (X)_K
\]

(9)

\[
P \equiv K \rightarrow P, P \equiv (X)_K
\]

(10)
Note that if $P \preccurlyeq X$ and $P \preccurlyeq Y$ it does not follow that $P \preccurlyeq (X, Y)$, since this would imply that $X$ and $Y$ were said at the same time.

The belief operator $| \equiv$ has certain necessary properties, such as if $P$ believes a set of statements if and only if $P$ believes each individual statement separately. It justifies the following rules:

\[
\begin{align*}
&P|\equiv X, P|\equiv Y \\
&P|\equiv (X, Y) \\
&P|\equiv (X, Y) \\
&P|\equiv X \\
\end{align*}
\]

(11)

(12)

(13)

A similar rule can also be applied to the operator $| \sim$:

\[
\begin{align*}
&P|\equiv Q| \sim (X, Y) \\
&P|\equiv Q| \sim X \\
\end{align*}
\]

(14)

Note that if $P| \equiv Q| \sim X$ and $P| \equiv Q| \sim Y$, it does not follow that $P| \equiv Q| \sim (X, Y)$, since this would imply that $X$ and $Y$ were said at the same time.

If one part of the formula is fresh, then the entire formula must be fresh:

\[
\begin{align*}
&P|\equiv \text{fresh}(X) \\
&P|\equiv \text{fresh}(X, Y) \\
\end{align*}
\]

(15)
3 Methodology

The key and most important for project development is using proper methodologies and choosing appropriate corresponding methods. This chapter will introduce the methodology and concrete methods used to accomplish the research work.

3.1 Knowledge Accumulation

Motivated by the realistic problems and challenges and to be able to have a complete understanding of the security issues in the Internet of Things as well as in P2P systems, state-of-the-art research work will be explored. At the same time, the basic principles, algorithms, protocols related to cryptography and security are also on the learning list, including the knowledge from textbooks and research papers. It is also essential to be able to distinguish the advantages and disadvantages of those cryptography algorithms and security protocols. It must be ensured that vulnerabilities in existing protocols are kept in mind and avoided for the proposed protocol. While those are not enough for implementation, as many recent realistic technical attacks against specific algorithms or protocols are not well introduced by books or exploited and summarized by papers, it is essential to summarized all kinds of vulnerabilities and attacks for each algorithm which could be used in the implementation.

3.2 Design Philosophy

Understand that SensibleThings platform is not only for implementation, but also to obtain the essence and philosophy of it. The design philosophy of this protocol is directly inherited from the SensibleThings platform. Drawing from the philosophy of this platform, the required features for the proposed protocol are listed as follows:

- Scalable - It should be scalable so that no central server is used for this protocol and no third party involved when communicating between two entities.

- Lightweight - It should be lightweight so that it can run on mobile devices with limited resources.

- Fast - Speed should also be guaranteed by avoiding extra handshakes for each session.
- Integritiy - Integrity of the messages is essential through encryption and signature against improper information modification.

- Authenticatable - The proposed protocol should also be authenticatable, which means enable to confirm the truth of an entity. This feature can be achieved through authorized certificates signed by an authority node.

- Flexible - Flexibility is should also be supported through a dynamic adjust mechanism other than redistribution.

- Available – Ensure timely and reliable access to and use of the information by timely and reliable transmission.

As a fully distributed platform, secure communication implemented on this platform should follow these principles as well. To add central servers for authentication is not permitted, which would not only imply bottleneck problems but would also limit the scalability of the whole system. Thus this principal leads to the design of a local storing mechanism for keys, certificates and so on. And aiming at different kinds of context in local storage, a hierarchical management is put forward in the implementation. Based on this, a mutual authentication mechanism is also born from the philosophy of this platform. This mechanism avoids the traditional authentication thorough a central server, but with a well signed certificate by one trustworthy entity in the platform. And to obtain such a valid certificate does require any extra procedure, but is accomplished simultaneously in an inevitable registration stage when an entity joins the platform.

### 3.3 Design Patterns

Before implementing the designed security protocol in the actual SensibleThings platform, it is important to completely understand the philosophy/principles and logic process of each action on the coding level. This essentially means to be aware of the design pattern and each module of the platform. According to [17], design patterns are a series of simple and elementary solutions to specific problems in object oriented software design. To increase the reuse and extensibility of the project, a well-designed pattern is essential. This will not only increases the cohesion, but also decrease the coupling of each of the module. Some patterns are used in this thesis work, like the abstract factory method.
3.4 Test-driven development

To increase the quality and efficiency of each module, a unit test framework is used. It is more than a framework that developers can use, but a kind of principle for developers to follow to avoid bugs and force each object to focus on one task. As one of the five object-oriented design principles — Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion (SOLID)[18] it suggests that every class should be responsible for one job (from the Single responsibility principle)[19]. These unit test frameworks are a core element of the Test Driven Development (TDD)[20]. The key rule of TDD is “test twice, code once”. This three-step procedure can also be called the TDD cycle. This thesis project is based on this framework, and the tool JUnit [21] is used to as a comprehensive test for the project.

3.5 Protocol Analysis

Being famous and pioneer in security protocol analysis, BAN Logic should no doubt be regarded as one of the most useful and efficient tools for analysis of current security protocols. After finishing the design of this protocol, BAN logic is used to analyze the proposed protocol and to repeat it with some revision. The deduction logic it established really helps to find the weaknesses in the protocol, verify the freshness and helps to remove the redundancy in the protocol. Before manipulating the logic for this protocol, a few essentials based on the realistic environment should be set up, which points out the direction of reinforcement when implementing.
4 Design and Implementation

This chapter will give a detailed description of the security protocol. It includes the philosophy, structure and communication. Then, the secure communication implemented on the SensibleThings platform will be introduced both its structure and each of its module.

4.1 Security Protocol

The core of the P2P security system for the Internet of Things is the security protocol. This protocol is the base of all systems’ communication and authentication. It determines the logic and framework of the system. The structure and design of this protocol mainly focus on high security, high efficiency and low cost.

4.1.1 Principles

As you may have noticed, the philosophy/art of this protocol has been briefly mentioned in the methodology part. Here, a more detail description will be presented. This actually acts as a core from which the whole logic of this protocol is derived.

Again, the fully distributed principal is the core part of this philosophy which promises no central point of failure. It also provides a high scalability feature to the system. Following this direction, the two main mechanisms serving this protocol are born. One is the mutual authentication through a well description certificate signed by a trustworthy authority node. Another is the local storage for sensitive information with a hierarchical management.

Usually, user authentication and sensitive information, such as password for an account, is achieved thorough a central server using online shopping as an example. This central management is simple and popular among online vendors. However, it is not suitable for a peer-to-peer distributed system, as it will result in a bottleneck problem limiting the scalability. So we have to think about how to authenticate an entity without a central server. Based on the original design feature for the platform, a stable bootstrap node provides an easy and quick way for a new node to join this platform. This could also be accomplished by talking to any other node in this platform. For security reasons, a trustworthy and stable node is a better choice. So then, this trustworthy node
could have the authority to permit others to come into its network. To a newcomer, the trust in the authority node is set before requesting to join.

To enable the ability to authenticate each other, a well description certificate signed by the trustworthy bootstrap node is essential. Because the bootstrap node is well accepted and trusted by others, it is possible for a common node to send a request to the bootstrap node to sign its certificate. This certificate contains a lot of information about the sender, such as the subject name, public key and most importantly the signature by the bootstrap node. With the signed certificate, each entity in this platform can authenticate others by verifying corresponding certificates and they are able to share a session key between them.

These certificate and session keys are so important that they can not be visited or modified by others. Certificates and public keys may be read by others, but cannot be modified by them. For session keys and private keys, it is more important that they are not seen by anyone except themselves. Hierarchical management is a good solution for the sensitive information.

4.1.2 Protocol Overview

As shown in Figure 8, there are two main parts in this security protocol: Registration, Join & Communication. This section will give a general introduction to each of them.

![Figure 8: Security Protocol Overview](image)

The registration is the first and most important stage for a new application join the system. At this stage, firstly the fresh application will first have to generate its a new key pair and a self signed root certificate. The root certificate is signed by the new private key, which is stored in a key store, called permanent key store (PKS). This permanent key store is protected with a password, possibly also by using other protection mechanisms. What is important is to make sure that content in the
permanent key store is not visible to un-permitted users or applications. The reason is that the permanent key store contains this extremely sensitive and import information, such as the private key, and others’ certificates, which are signed by the preset authority.

After the self-preparation when a new application enters into the platform, it will then generate a certificate signing request. During the registration, this request will be sent to an Authority Server (AS) / Authority Node (AN). When received, the server will handle this request by signing it with its private key. And then transmit it back with the server’s self-signed root certificate. During all these communications, a mutual session key is negotiated before the request. So the following communications will be encrypted by this session key.

The second stage is join & communication. At this stage, applications can directly join the P2P platform if they have registered before. Normally, an identifier is required in this platform to maintain a distributed hash table (DHT). So that based on this overlay, each entity in this platform can find each other through the identifier. After that, communication between tow nodes occurs. Depending on the certificate or session key the entity has or does not have, there are three situations in which an entity will execute different procedures, which will be discussed later on.

4.1.3 Procedure I: Registration

Figure 9 illustrates, the registration process carried out between the recently joined client and the Authority Node (AN). At the beginning, normal communication connection will be established among the entities in the platform, such as the Transmission Control Protocol (TCP) or Reliable User Datagram Protocol (RUDP). When the client starts to register to the AS, an SSL communication connection request will first be sent to the AS. After this, the following communication will be based on the SSL. This step may be not necessary, but it can enhance the security of this registration transaction.
Then the client will send a registration request message to the AS. Not only the request, but also the identity together with a nonce have been sent to the AS. The nonce is used to protect from the replay attacks that ensure that old communication cannot be reused.

When the AS received the registration request, it will send back a registration reply message (Shown in Figure 11), which includes an AS self-signed certificate and a nonce signed by the AS. This signature is important because it cannot only be used to check the freshness of the nonce from the latest request, but also to verify the authentication of the response.
Figure 11: Registration Response Message

After receiving the registration reply from the authority node and passing the verification of the certificate and nonce, the client will store the certificate from the AS in the local permanent key store as it contains the public key of the AS. Then a certificate signing request will be generated including the identity, public key, and signature of the client involved. Finally, a certificate signing request message will be transmitted to the AS, in addition to the certificate signing request, another nonce encrypted using the public key of the AS included. This message structure is shown in figure 12.

Figure 12: Certificate Signing Request Message

When receiving the certificate signing request message from the client, the AS will first check the identity and signature of the certificate signing request. If they are all valid, the AS will generate an issued certificate through the request by setting the issuer as AS and finally sign it. After that, a session key will be generated to apply a symmetric encryption for the next communication. This session key will also be encrypted when transmitted to the client. Besides, the received nonce together with a new nonce, and the root certificate will all be encapsulated into a certificate signing response message. The detailed structure of the certificate signing response message is shown in Figure 13.
Finally, when the client received the certificate signing response message, the negotiation between them is almost complete. Of course, checking the identity, signature and nonce is necessary. The encrypt session key will then be stored in a local temporary key store. So if needed, the following communication between the client and the AS will be protected using the session key. Finally, a certificate accepted message is sent back to the AS, together with an encrypt nonce received from the AS. The structure of certificate accepted message can be seen in figure 14. After receiving the certificate accepted message, if there are no more registration transactions to the AS, both of them will shift SSL communication back to the original communication, otherwise only the client will change back.

4.1.4 Procedure II: Join & Communication
As mentioned in 4.1.1, there are three situations in the second procedure, all of which can be seen in Figure 15. Depending on whether the certificate and the session key is shared with the receiver, sender should carry out different actions according to different situations.
Figure 15: Security Protocol Procedure II: Join & Communication

As shown in figure 15, the worst scenario is situation 1 in which the certificate of the client to whom the sender would like to send is invalid, e.g. expired. Neither sender has the corresponding certificate. So the sender would first have to exchange the certificate. Situation 2 is more positive in that sender has the valid corresponding certificate of the target. However, they have not shared a valid session key (e.g. symmetric key), nor has the session key expired. It is then essential for them negotiate a session key. The most positive situation is when sender finds that it has a valid session key with a corresponding receiver. So that it is able to encrypt the message directly and send it out.
Figure 16: Security Protocol Procedure II: Join & Communication (Situation 1)

Figure 16 shows the most positive situation when the clients start to communicate with each other. Because the key they have is valid and they can use it directly without any redundant communication.

![Security Protocol Procedure II: Join & Communication (Situation 1)](image)

Figure 17: Security Protocol Procedure II: Join & Communication (Situation 2)

At time passes, the session key between two entities will expire. Neither sender has communicated with the destination recently. Both of these two situations require the sender negotiate a session key with the target. Fortunately, sender has the valid signed certificate of the target, which means they actually contact before. The above description is situation 2, which can be seen in Figure 17. In this situation, Client 1 wants to send a message to client 2. While it discovered that there is no valid key in the local temporary key store, it did find a valid certificate. So the up-going message will be stored temporarily until they have a valid session key. After two session key exchange messages, both of them have the session key which could be used in a symmetric cipher. To make sure that target, Client 2, can authenticate the session key exchange message from Client 1, a signed certificate from Client 1 will be sent together with the request message. Because it may be that Client 2 will find that the certificate of Client 1 has expired, or lost it. Finally, the sender uses the session key to encrypt the message and sends it out.
At the very beginning, with the exception of the authority node, no entity has been in touch with another. This is described in the situation 3 (shown in Figure 18). As no two entities have been in contact before, they do not have each other’s certificates. So before client 1 sends out the secure message, a valid certificate on behalf of them must be shared. After this step, situation 3 becomes situation 2. The entities will act as in situation 2. Finally, they will use the session key they negotiated to encrypt the message.

4.2 Secure Communication in SensibleThings Platform

In this section, a detailed discussion about the protocol implemented on the SensibleThings platform will be presented. As a security protocol, it is natural to think about implementing this security protocol as a kind of communication, such as SSL, so that it can be transparent to the applications, and application developers can easily use this secure communication to protect their information without worrying about the details of this security protocol.

4.2.1 General Structure

In this section, the general structure of the secure communication based on the SensibleThings platform will be introduced. As the structure of
SensibleThings platform was mentioned in section 2.6, Figure 19 shows of the general structure of this platform.

Figure 19: Position of Secure Communication implemented on SensibleThings platform

Figure 19 shows the position in which this secure communication implemented on this platform. Originally, other from communication have been implemented, such as TCP, RUDP, SSL. Drawing from the knowledge of these existing communication protocols, developing this new means of communication takes little time. Because these existing communication mechanisms provide a good demonstration of how to implement the communication interfaces.

Figure 20: General Structure of Secure Communication
In Figure 20, the general structure of secure communication is presented. Considering the existing IO communications have been implemented based on the Java socket, there is little point in implementing a new communication for secure communication. On the other hand, this shows the lose coupling of each module in this platform that makes it easier to scale and increase the reusability of existing modules. Based on this kind of architecture and design philosophy, secure communication is developed.

In general, secure communication can be separated into three levels. The bottom level is the communication level. In order to easily distinguish the secure communication, the bottom level will be called the IO Communication. The bottom level is mainly responsible for the data stream transmission. It handles things such as the acknowledgement of received packets, retransmission of lost packets and other things that are the responsibilities of communication. Fortunately, these reliable complex transmission mechanisms are implemented in the standard Java Development Kit (JDK), which provided the APIs for developers to call.

The second level is the main level that we focus on. All messages pointed out in the previous protocol designing 4.1 are implemented in this part. Message Handler not only handles all the messages serving this protocol, but also needs to capsulate all messages transmitted from up levels, such as the synchronous messages from the lookup service. In addition to the message handler, there is another component called security configuration. As the name has suggests, it is a configuration severing secure communication. It includes many parameters that secure communication, such as what the security level applications would like to use, where to store the key store files and so on. Details on this part can be found in 4.2.2.

The last module is responsible for more trivial operations, like encryptions, decryptions, storing the keys, drawing the keys etc. This module is called security manager. It provides all specific operations that secure communication needs. Real operations are not carried out by the security manger, but by its "staffs"- a few specialised small modules focusing on one area only. The security manager is just like a governor that includes all services from different artisans, and provides a series of uniform and complex services for upper levels. Details can be found in 4.2.3.
4.2.2 Secure Communication Implementation

First, security configuration will be described in detail in this section. As many parts in this protocol implementation are related to this configuration, it is best to introduce it first.

As shown in Figure 21, there are mainly three parts in this configuration: bootstrap, key store and security level. Bootstrap is an administrator managed stable node that is used in order for new nodes to be able to join quickly. The meaning of the node may be a little ambiguous. Basically, it can be anything, such as a computer, a mobile phone or one set-top-box. It is also possible to set up a bootstrap node by yourself. So it is essential to specify the identity and address of the bootstrap. Currently, a UCI of the node is used in this platform, with the assumption that the UCI is unique for each node in this platform.

The second part is the setting of the key store. The identity, or the file name or the key store should be set. As it is possible that a node could login through different UCIs, it is essential for the to name to be a unique identity for the key store file. Another issue is to set the directory of the key store, telling the system where to store the key store file.

The last part of the security configuration is the security level. This security level provides above applications different choices of security level, which could be suitable to different situations. The higher the level is, the more secure the communication will be. It means the higher security level uses longer key to encrypt the messages and the corre-
sponding lifetime will be shorter. So if a higher security level is adopted, it will devote more time to encrypt and decrypt messages. The “description” is a short introduction of this security level. In the symmetric/asymmetric cipher part, the algorithm, mode, key length and the lifetime are needed. The signature part only requires the algorithm it plans to use. A balance between security and efficiency should be considered. Either way, it is all up to developers to choose a suitable level for their applications.

As mentioned above, the security handler is responsible for the router of all the messages going through by this protocol. Mostly those message mentioned in 4.1 will be routed to the security manager, which will be discussed later on. The kinds of messages can been from the following list.

- Communication Shift Message: Send the signal to inform the Bootstrap node to shift to another communication. In this work, nodes will only shift between RUDP communication and SSL communication.

- Registration Request Message: Send the registration request to the bootstrap node from the recently arrived nodes.

- Registration Response Message: After receiving a registration request, the bootstrap node will send back a self-signed root certificate contained in the registration response message. So that they can set up a secure communication by using one asymmetric cipher.

- Certificate Request Message: By using the public key from the registration response message, the node sends a certificate request message including a certificate signing request, encrypted ID and encrypted nonce. Included in the certificate signing request, you will find its public key, the lifetime of the key, ID, and their signature.

- Certificate Response Message: After verifying the certificate request, the bootstrap node signs an issued certificate. At the same, it starts to set up a session key between them. The key is contained in the certificate response message, and of course encrypted by one asymmetric cipher. In the message, a new nonce plus
the previously received nonce are sent back to maintain the freshness of the message.

- Certificate Accepted Message: The node sends the received nonce back encrypted by the new session key. This information make up the certificate accepted message.

- Session Key Exchange Message: If one node wants to send messages to another and it only finds a valid certificate for the node, a session key exchange message will be sent first to establish a symmetric encryption between the nodes. In addition to a session key, a nonce is also encrypted in this message. An issued certificate is also attached to this message used for authentication.

- Session Key Response Message: After verifying the session key exchange message, the target sends back a session key response message. This message includes a new encrypted nonce and a received nonce to maintain the freshness of the message.

- Certificate Exchange Message: If one node wants to send messages to another, and it finds that there is no valid certificate the node, a certificate exchange message will be sent out to share their issued certificates from the bootstrap node. The certificate of the sender is included in this message.

- Certificate Exchange Response Message: When receiving the certificate exchange message, verification of this message will first be carried out. After verification, it sends back a certificate exchange response message. This message includes the issued certificate and an encrypted received nonce.

- Secure Message: This type of message is used to serve the upper levels. All upper level messages will be encapsulated in the secure message. Information is encrypted in this payload, and the corresponding signature is linked to the message.

### 4.2.3 Security Manager

The security manages is a collection of services for the message handler. It is like a controller which processes all the coming and going information contained in the messages. It also manages the local key store and provides all the cryptography related interfaces.
Figure 22 illustrates the structure of the security manager. In terms of modules, there are four parts: asymmetric encryption, symmetric encryption, certificate and signature. Different modules have different operations. In the asymmetric/symmetric module, there are three kinds of operations: encryption, decryption and keys generation. In the certificate module, there are also three operations: certificate generation, verification and issuing. In the signature module, there are only two operations: sign and verify. All these operations frequently interact with the key store: storing keys and certificates and fetching them from the key store.
5 Results

This chapter will introduce the analysis of the secure communication based on its efficiency. More importantly, the security protocol will be presented in this chapter.

5.1 Test environment

To test the efficiency of the secure communication, two nodes in the Local Area Network (LAN) are established.

![Structure of Test Environment](image)

Figure 23: Structure of Test Environment

As shown in Figure 23, two nodes running on two computers are set up. These two computers are running 32-bit Windows 7 with the hardware environment (Intel(R) Core(TM) 2 Duo CPU E8400 3.00GHz, RAM: 4.00GB). All applications are running on the Eclipse J2EE platform.

5.2 Performance

To test the performance of the secure communication, we began by testing different communication encryption algorithms and different encryption modes of symmetric cipher. There are two main symmetric ciphers preferred in this system: AES and RC4. As AES is a block encryption cipher, there are six different modes that could be used: Cipher-block chaining (CBC), Electronic codebook(ECB), Cipher feedback(CFB), Output feedback(OFB), Propagating cipher-block chaining(PCBC) and Counter(CTR).

As shown in Figure 24, the scenario to test the performance of different symmetric encryption is to encrypt a random generated message and
then decrypt it. The time of the whole process is measured by nanoseconds. The exact time is measured by making 10,000 tests and having the average returned.

![Performance test Scenario](image)

Figure 24: Performance test Scenario

Figure 25 shows the result of the performance test 128 bits key. The x-axis is the length of the randomly generated text in bytes. The y-axis is the average time of each test in nanoseconds. From the figure, we can see that there is little difference when the length of the text is around 10 to 10,000 bytes. Normally RC4 is the most efficient. When the text length is increased to 100,000, ECB mode is the fastest by far. The following are CBC and RC4. The last four is CFB, OFB, CTR and PCBC. The reason for the experimental result can also be explained by its principles.

![Symmetric Encryption (128 Bits)](image)

Figure 25: Symmetric Encryption (128 Bits)

Figure 25 shows the result of the performance test using a 192 bits key. Between 10 to 10,000, RC4 still occupies the leading position. The remaining are almost same as in Figure 25. When increasing to 100,000 bytes, ECB is also the fastest, as shown in Figure 25.
Similarly, Figure 27 shows the result of the performance test using a 256 bits key. These is little difference when compared to the previous two figures, except CBC shows a small advantage when the text becomes 100,000 bytes long.

![Symmetric Encryption (192 Bits)](image1)

**Figure 26: Symmetric Encryption(192 Bits)**

![Symmetric Encryption (256 Bits)](image2)

**Figure 27: Symmetric Encryption(256 Bits)**

### 5.3 Comparison with SSL

A SSL communication has been built into this platform in the previous development stage of the SensibleThings platform. It used a cipher, which has the standard name “SSL_DH_anon_WITH_RC4_128_MD5”. It means this SSL cipher uses the Diffie-Hellman(DH) algorithm for key
exchanges, the 128 bits key used in the RC4 and MD5 is the digest algorithm. “anon” stands for anonymous, which is a cipher each node can use without pre-sharing the certificates. For academic research, we assume that there is no man-in-the-middle attack against DH in this platform. To compare, the secure communication uses the same block cipher algorithm.

The test scenario is shown in Figure 23. One node will generate a random test with specified length and transmit it to another node with these two protocols respectively. When node 2 receives the message, it will send it back as soon as possible. Because there is no synchronization between these two nodes, the total transmission time is only measured on one node.

Figure 28 demonstrates the comparison between SSL and the security protocol. Each result is the average of 1,000 transmissions with certain intervals between two transmissions. The x-axis is the length of each generated message in byte, and the y-axis is the measured transmission time. From the figure, we can see that the proposed security communication (green line) is more efficient than SSL. In average, SSL will take five times longer than the proposed protocol.

![SSL vs. Security Protocol](image)

**Figure 28: SSL vs. security protocol**

### 5.4 Security Protocol Analysis

First, the protocol is introduced below in a simplified form. A denotes a principal, S the authentication server. $K_A$, $K_S$ are the public keys of A and S, respectively. $K_A^{-1}K_S^{-1}$ are the secret keys that matches $K_A$ and $K_S$. The server S generates $K_{AS}$, which becomes the session key between A and
A Peer to Peer Security Protocol for the Internet of Things
– Secure Communication for the SensibleThings Platform

Results
2014-10-13

S. 𝑁_𝑎, 𝑁_′_𝑎, 𝑁_𝑠, 𝑁_′_𝑠 are all the nonces that were generated by A and S respectively.

1. A → S:  
   \[ A, 𝑁_𝑎 \]

2. S → A:  
   \[ S, 𝑘_𝑠, \{ A, 𝑁_𝑎, 𝑁_𝑠 \} 𝑘_𝑠^{-1} \]

3. A → S:  
   \[ A, \{ A, 𝑘_𝑎, 𝑁_′_𝑎 \} 𝑘_′_𝑎, \{ A, 𝑘_𝑎, 𝑁_𝑠 \} 𝑘_″_𝑎^{-1} \]

4. S → A:  
   \[ S, \{ A, 𝑘_𝑎, 𝑁_′_𝑎 \} 𝑘_′_𝑎, \{ A, 𝑁_′_𝑎, 𝑁_′_𝑠, \{ A, 𝑘_𝑎, 𝑘_𝑠 \} 𝑘_″_𝑎^{-1} \} 𝑘_𝑎, \{ A, 𝑁_′_𝑎, 𝑁_′_𝑠, \{ A, 𝑘_𝑎, 𝑘_𝑠 \} 𝑘_″_𝑎^{-1} \} 𝑘_𝑎 \]

5. A → S:  
   \[ \{ 𝑁_′_𝑠 - 1 \} 𝑘_𝑎 \]

A sends a request to S only its identity and a nonce. S replies to A with its public key, and encrypts the request with its secret key. A checks the nonce and then sends enough information including its public key and a new nonce, and both of them are encrypted with S’s public key, and A’s secret key separately. Then S sends back a session key encrypted with A’s public key, adding more information embedded in an encrypted message by the session key. Finally, the use of 𝑁_𝑠 – 1 in the last message is conventional. Almost any function of 𝑁_𝑠 would do, as long as S can distinguish his messages from A’s.

Now we transform the protocol. The idealized protocol is as follows:

1. Omitted

2. S → A:  
   \[ \{ 𝑘_𝑠 \} 𝑘_″_𝑎^{-1} \to S, \{ A, 𝑁_𝑎, 𝑁_𝑠 \} 𝑘_″_𝑎 \]

3. A → S:  
   \[ \{ A, 𝑘_″_𝑎 \} 𝑘_″_𝑎^{-1}, \{ A, 𝑘_″_𝑎 \} 𝑘_″_𝑎^{-1} \]

4. S → A:  
   \[ \{ < A, ( A 𝑘_𝑎 → S ) \} 𝑘_″_𝑎 \}

5. A → S:  
   \[ \{ 𝑁_″_𝑠 \} 𝑘_″_𝑎 \]

The first message is omitted, since it does not contribute to the logical properties of the protocol. Message 2 and 3 are straightforward. But the following two messages need some explanation. In message 4, 𝑁_′_𝑎 is used as secret, so the < 𝑋 > 𝑘 notation is used. In message 5, the session key does not show in the original message. This interpretation of the session message actually shows that A wants S to believe that it has already got
the session key. At this point, we can convince ourselves that the idealized protocol is an exact representation of the actual protocol.

To start analysing, we begin with some assumptions:

\[ A \equiv K_s \rightarrow S \]

\[ A \equiv K_a \rightarrow A \]

\[ S \equiv K_s \rightarrow S \]

\[ S \equiv K_a \rightarrow A \]

The upper group four is about the public key that each principal knows, that is the public key of the agent S, as well as its own keys.

\[ S \equiv A \leftarrow K_{as} \rightarrow S \]

Also, S believes the session key that was generated (shown in the upper formula).

\[ A \equiv (S \Rightarrow K_s \rightarrow S), A \mid \equiv (S \Rightarrow A \leftarrow K_{as} \rightarrow S), A \mid \equiv (S \Rightarrow A, S, \Rightarrow K) \]

\[ S \equiv (S \Rightarrow A \leftarrow K_{as} \rightarrow S), S \mid \equiv (A \Rightarrow K_a) \]

The upper group of five indicates the trust that A and S have in one another to generate a good encryption key and session key, and S can honestly sign the certificate request from A (the third assumption in this group).

\[ A \equiv fresh(N_a), A \equiv fresh(N_s), S \equiv fresh(N_a), S \equiv fresh(N_s) \]

The final four assumptions show that four nonces have been invented by various principals, and it also shows who considers them to be fresh.

Apply the postulate on message meaning, nonce verification, jurisdiction, we get the following final beliefs. The detailed deduction of these beliefs can be seen in Appendix A:

\[ A \equiv S \equiv (A, N_a, N_s), S \equiv A \equiv K_a \rightarrow A \]
\[ A| \equiv (A \leftarrow^{k_{as}} S) \]

\[ A| \equiv (\{A, S, ^{K}A\}_{K_{S}^{-1}}) \]

\[ S| \equiv A| \equiv (A \leftarrow^{k_{as}} S) \]

With a series of deduction following the logic, we got that A believes the session key shared between itself and S. More importantly, we got that, S believes that A is convinced of the session key, and A believes the authorized certificate signed by S.

### 5.5 Secure Communication Analysis

This analysis is made from the secure communication point of view if this project. It is not a protocol analysis, which focuses on the fundamental logic of the protocol, rather this analysis is more practical and maybe more complex, as it concerns more technical details concerning secure communication. Another task for this analysis is to point out the potential risk or limitations of the implementation, which could also been included in future work.

Considering the implementation of a security protocol, the main and most important job is still in cryptography. Almost all cryptography principles is public and could be implemented on their own. While selecting existing libraries could be better than making your own, not only save lots of time and energy, but also, these libraries are well tested and maintained by security communities. Although there is certain risk involved, such as the Heartbleed bug, choosing released libraries still has more advantages then disadvantages considering the limited time and knowledge.

The Java cryptography architecture is a great tool that helps to build almost 90 percent of the whole project. Although its APIs are well documented, they still look like black boxes to the outside which limits the extensibility of many functions. Once these is a potential bug in these libraries, it will not be quickly solved, as these are not open sourced. This may be omitted in this project. During implementation, a bug was found in the key store library. The solution taken is to avoid this function and mark it clearly in the code. The remaining 10 percent is completed with the help of Bouncy Castle libraries [16], which has better support for certificate operations. However, the Bouncy Castle crypto APIs are poorly documented, which tends to confuse developers. The
reason for adapting these libraries is that the APIs from the Java cryptography architecture for certificates are old-fashioned and many are deprecated.

Another issue that should be pointed out here is the lifetime of the session key. This parameter could be set through the security configuration, which has been mentioned above. This parameter is actually depends on the application requirements. However, developers should carefully consider this, because there is a balance between its security and efficiency. That means that if the lifetime of the session key is too long, then it more likely to be compromised. If it is too short, a new session key will be exchanged frequently, which will make the system heavier.
6 Conclusions

The Internet of Things has been enriching people’s life by facilitating social aspects of life, smart choices, better safety, and more. Its unlimited development space is makes the social blueprint colorful. However, with the growing complexity and increasing threats in security, it becomes a significant challenge for systems and applications to set up a security mechanism, which provides an efficient protection for the communication traffic. We proposed a peer-to-peer security protocol to address this problem. A corresponding secure communication is implemented on the SensibleThings platform.

The required features of this protocol are achieved through a series of mechanisms. It is scalable as no central server is used and no third party involved when communicating between two entities. It is also light-weight enough as keys and certificates are stored in a well protected lightweight key store instead of in a heavy database, which is able to run mobile devices with limited resources. It is much faster than the SSL cipher, which was previously adopted by the platform. This is because extra handshakes are avoided in each session. A dynamic security level adjustment mechanism makes it flexible as well, which could satisfy different application requirements. Integrity is guaranteed by being good encrypted and signatures. The authenticatable character is achieved through the authorized certificate signed by the authorized node. The information is timely available through fast and reliable transmission. This protocol follows the philosophy of the SensibleThings platform exactly, which makes the platform more secure.

For the implementation of secure communication, an abstract factory pattern is adopted to increase the cohesion of each component and decrease the coupling. A template method pattern is used for basic functions, which increases the extensibility of the secure communication. Each basic function is well achieved through test-driven development method, which enhances the robustness of this secure communication.

To evaluate and analyze the security protocol, BAN logic is used as an analysis tool to find the weakness of the proposed protocol and help to remove the redundancy without weakening it. The analysis result shows that with few assumptions, it is able to meet the security requirement. In addition, a performance comparison test with SSL is executed based on the SensibleThings platform. The test result shows
that proposed security is much more efficient than SSL by avoiding unnecessary handshakes.

This thesis work provides an efficient security solution for the systems or platforms severing the Internet of Things. The implemented secure communication greatly improves the protection of the communication in the SensibleThings platform. Similar secure communication can also be built for other systems. Equipped with this protection, people's privacy is well protected in the Internet of Things. It also promotes the development of the Internet of Things. Based on this work, systems for the Internet of Things could be applied to more areas and provides safer and more convenient applications to the world.

6.1 Future Work

Our future work will be to explore the key features of security, as well as other important issues concerning the authentication, permissions management, password management, malicious node detection and prevention and serious attack tests against this secure communication. Among them, password management and recovery is particularly important which is the door of privacy. It could be done in many ways, like password resetting emails, text messages and etc. However, which of them is better and more secure still needs to be thoroughly researched.

The permission management for this secure communication could be achieved by setting different policies through the Java authentication and authorization services. Important and sensitive files like the key store files are required for particular protection.

Malicious node detection may be a little more complicated but it is not impossible. Some solutions based on corresponding protocols have been published. These may provide much useful information about this problem.

To test serious attack for this implemented secure communication may need professional tool and experience. Some well known tools, like an advanced penetration testing Linux distribution, Kali Linux, are freely and easily available. However, the difficult part is the strategy adopted for the testing. It requires considerable experience and knowledge.
7 References


A Peer to Peer Security Protocol for the Internet of Things – Secure Communication for the SensibleThings Platform

References

2014-10-13


[20] P. Hamill, Unit test frameworks, Sebastopol, Calif. : O’Reilly, 2004

A Peer to Peer Security Protocol for the Internet of Things – Secure Communication for the SensibleThings Platform

References

2014-10-13


Appendix A: Logical reasoning process for security protocols (Procedure I) with BAN logic

Simplified protocol
1. $A \rightarrow S$: $A, N_a$
2. $S \rightarrow A$: $S, K_s, \{A, N_a, N_s\}_{K_s^{-1}}$
3. $A \rightarrow S$: $A, \{A, K_a, N'_a\}_{K_a}, \{A, K_a, N_s\}_{K_a^{-1}}$
4. $S \rightarrow A$: $S, \{A, K_{as}, N'_a\}_{K_{as}}, \{A, N'_a, N'_s, \{A, S, K_a\}_{K_a^{-1}}\}_{K_{as}}$
5. $A \rightarrow S$: $\{N'_s - 1\}_{K_{as}}$

Idealization of protocol
1. Omitted
2. $S \rightarrow A$: $S \rightarrow \{A, N_a, N_s\}_{K_s^{-1}}$
3. $A \rightarrow S$: $\{A \rightarrow A, N'_a\}_{K_a}, \{A \rightarrow A, N_s\}_{K_a^{-1}}$
4. $S \rightarrow A$: $\{< A, (A \leftarrow K_{as} \rightarrow S) >_{N'_a} \}, \{A, N'_a, N'_s, \{A, S \rightarrow A\}_{K_a^{-1}}\}_{K_{as}}$
5. $A \rightarrow S$: $\{N'_s, (A \leftarrow K_{as} \rightarrow S)\}_{K_{as}}$

Assumptions for protocol

$A \equiv (K_s \rightarrow S), A \equiv (K_a \rightarrow A), S \equiv (K_s \rightarrow S), S \equiv (K_a \rightarrow A)$

$A \equiv (S \rightarrow A), A \equiv (S \rightarrow A), S \equiv (A \leftarrow K_{as} \rightarrow S), A \equiv (S \rightarrow A, S \rightarrow A)_{K_a^{-1}}$

$S \equiv (S \rightarrow A \leftarrow K_{as} \rightarrow S), S \equiv (A \leftarrow K_{as} \rightarrow S), S \equiv (A \leftarrow K_{as} \rightarrow S)$

$A \equiv fresh(N_a), A \equiv fresh(N'_a), A \equiv fresh(N_s), S \equiv fresh(N'_s)$
Reasoning process for protocol

1. Message 1 is omitted, because it has no contribution to the logical properties of the protocol.

2. From message 2: $S \rightarrow A: \rightarrow S, \{A, N_a, N_s\}_{K_a^{-1}}$, we can easily get:

   $$ A \triangleleft \{A, N_a, N_s\}_{K_a^{-1}}. $$

   From the assumptions we get $A| \equiv \left( \rightarrow S \right)$. Then according to postulate 1 in section 2.8.3, we deduce:

   $$ A|\equiv\left(\rightarrow S\right)_{A\leftarrow\{A, N_a, N_s\}}K_a^{-1} $$

   so we get $A| \equiv S| \sim \{A, N_a, N_s\}$.

   From the assumptions, we get $A| \equiv fresh(N_a)$. According to postulate 15 in section 2.8.3, we obtain: $A| \equiv fresh(A, N_a, N_s)$.

   Because $A| \equiv fresh(A, N_a, N_s)$ and $P| \equiv Q| \sim \{A, N_a, N_s\}$, according to postulate 4 in section 2.8.3, we get: $A| \equiv S| \equiv (A, N_a, N_s)$.

3. From message 3: $A \rightarrow S: \{A, \rightarrow A, N_a, N_s\}_{K_a'^{-1}}, \{A, \rightarrow A, N_s\}_{K_a^{-1}}$, we get:

   $$ S \triangleleft \{A, \rightarrow A, N_s\}_{K_a^{-1}}. $$

   From the assumptions, we get $S| \equiv \left( \rightarrow A \right)$. Then according to postulate 1 in section 2.8.3, we deduce:

   $$ S|\equiv\left(\rightarrow A\right)_{S\leftarrow\{A, \rightarrow A, N_s\}}K_a^{-1} $$

   so we get $S| \equiv A| \sim \left( A, \rightarrow A, N_s \right)$. Finally, according to postulate 13, we get $S| \equiv A| \equiv \left( A, \rightarrow A, N_s \right)$.

4. From message 4: $S \rightarrow A$:
Because $A| \equiv fresh(N_a')$, according to postulate 15 in section 2.8.3, we obtain: $A| \equiv fresh(A \leftarrow K_{as} \rightarrow S)$.

Because $A| \equiv fresh(A \leftarrow K_{as} \rightarrow S)$ and $A| \equiv S| \sim (A, A \leftarrow K_{as} \rightarrow S)$, according to postulate 4 in section 2.8.3, we get $A| \equiv S| \equiv (A, A \leftarrow K_{as} \rightarrow S)$.

From the assumptions, we get $A| \equiv (S| \Rightarrow A \leftarrow K_{as} \rightarrow S)$, and $A| \equiv S| \equiv (A, A \leftarrow K_{as} \rightarrow S)$, according to postulate 5 in section 2.7.2, we get $A| \equiv (A, A \leftarrow K_{as} \rightarrow S)$. Then according to postulate 12, we get $A| \equiv (A \leftarrow K_{as} \rightarrow S)$.

Because $A \leftarrow \{A, N_a', N_s', \{A, S, \rightarrow A\}_{K}\}_{K_{as}}$ from the message, and $A| \equiv (A \leftarrow K_{as} \rightarrow S)$, according to postulate 1, we deduce: $A| \equiv S| \sim (A, N_a', N_s', \{A, S, \rightarrow A\}_{K}\}_{K_{as}}$.

Because $A| \equiv fresh(N_a')$, according to postulate 15 in section 2.8.3, we obtain: $A| \equiv fresh(A, N_a', N_s', \{A, S, \rightarrow A\}_{K}\}_{K_{as}}$.

Because $A| \equiv fresh(A, N_a', N_s', \{A, S, \rightarrow A\}_{K}\}_{K_{as}}$ and $A| \equiv S| \sim (A, N_a', N_s', \{A, S, \rightarrow A\}_{K}\}_{K_{as}}$, according to postulate 4 in section 2.8.3, we get: $A| \equiv S| \equiv (A, N_a', N_s', \{A, S, \rightarrow A\}_{K}\}_{K_{as}}$. Then according to postulate 13, we get $A| \equiv S|\equiv (\{A, S, \rightarrow A\}_{K}\}_{K_{as}}$.

From assumption $A| \equiv (S| \Rightarrow \{A, S, \rightarrow A\}_{K}\}_{K_{as}}$, and $A| \equiv S| \equiv (\{A, S, \rightarrow A\}_{K}\}_{K_{as}}$, according to 5 in section 2.7.2, we get $A| \equiv (\{A, S, \rightarrow A\}_{K}\}_{K_{as}}$.

5. From message 5 $A \rightarrow S$: $\{N_s', (A \leftarrow K_{as} \rightarrow S)\}_{K_{as}}$, we get: $S \leftarrow \{N_s', (A \leftarrow K_{as} \rightarrow S)\}_{K_{as}}$. From assumptions, we have $S| \equiv A \leftarrow K_{as} \rightarrow S$.

Then according to postulate 1, we get $S| \equiv A| \sim (N_s', A \leftarrow K_{as} \rightarrow S)$. 

References

A Peer to Peer Security Protocol for the Internet of Things

– Secure Communication for the SensibleThings Platform

2014-10-13
Again from the assumptions, we have $S| \equiv fresh(N_s')$. According to postulate 15 in section 2.8.3, similarly, we obtain: $S| \equiv fresh(N_s', A \leftarrow^{K_{as}} S)$.

Because $S| \equiv fresh(N_s', A \leftarrow^{K_{as}} S)$ and $S| \equiv A| \sim (N_s', A \leftarrow^{K_{as}} S)$, according to postulate 4 in section 2.8.3, we obtain: $S| \equiv A| \equiv (N_s', A \leftarrow^{K_{as}} S)$. Then according to postulate Formula, we get $S| \equiv A| \equiv (A \leftarrow^{K_{as}} S)$.