Privacy preservation in Internet of Things:

A secure approach for distributed group authentication through Paillier cryptosystem

Elia Aielli

Department of Information Engineering

University of Pisa

Master degree in Telecommunication Engineering

Academic Year 2013/14

I would like to dedicate this thesis to my loving family and my beautiful girlfriend...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Elia Aielli

Academic Year 2013/14

Abstract

The IoT is a wonderful concept in continuous evolution; in my thesis work I first introduced the key concepts on which my work is based, like privacy and security. In the second chapter I present an overview of the IoT state of art updated at April 2015, knowing that one day to another there is the chance to discover new concepts and revolutionize all the old thinking. In the third chapter I explained the theoretical background,necessary to understand the different components of the cryptosystem proposed in my work. In the fourth chapter I described briefly which scenarios my application can fit (with a short description of what a Mobile Ad-hoc Network is) and explained my thesis work in depth, presenting the project part and the implementation one, and analyzing the results obtained by simulations on the test bed. At the end I drew some concluding remarks on my work.

Table of contents

List of figures

Chapter 1

Introduction

"Security is always excessive until it's not enough."
Robbie Sinclair

With the advent of new technologies, social networks and portable devices with Internet access, people's privacy protection is a key concept to be faced with. Nowadays people don't understand how many personal information they give to whoever knows how to move on the web. New laws and certifications are created and deployed to protect citizen's security, but only a better awareness of risks can really protect the "medium" user. Therefore, it is essential to let the people know those kinds of risks and to protect them in order to avoid facing problems the hard way. Portable objects have made possible the realization of the Internet of Things, literally an environment where every little object has a possibility to communicate his data electronically and can build a net with his counterparts to improve or simply to facilitate people's life, at the cost of their own privacy. My application wants to give a possible solution in order to make people able to communicate in an environment with constrained resources, in the safest, fastest and most efficient way that I know, but the awareness of an imminent attack is always the best weapon in the user's hands.

Chapter 2

Internet of Things

2.1 The Internet of Things

The Internet of Things is an emerging global Internet-based technical architecture facilitating the exchange of goods and services in global supply chain networks. With the exchange of an incredible amount of information, it has an huge impact on the security and privacy of the involved stakeholders.
To analize all the problematics involved with this architecture, we must highlight the fact that being a fusion of heterogeneous networks, it brings with himself the problematics of old technologies on which it sets up, like Internet, sensor networks and mobile communication networks. Furthermore it has to take into account the new problems growing with the IoT architecture itself, like heterogeneous network authentication, informa-

tion store and management, access control and last but not least privacy protection.
By the fact that actually doesn't exist an official and unificated IoT Architecture, in this thesis work we'll use a 3-layered structure [29], evidenced in Figure 2.1.

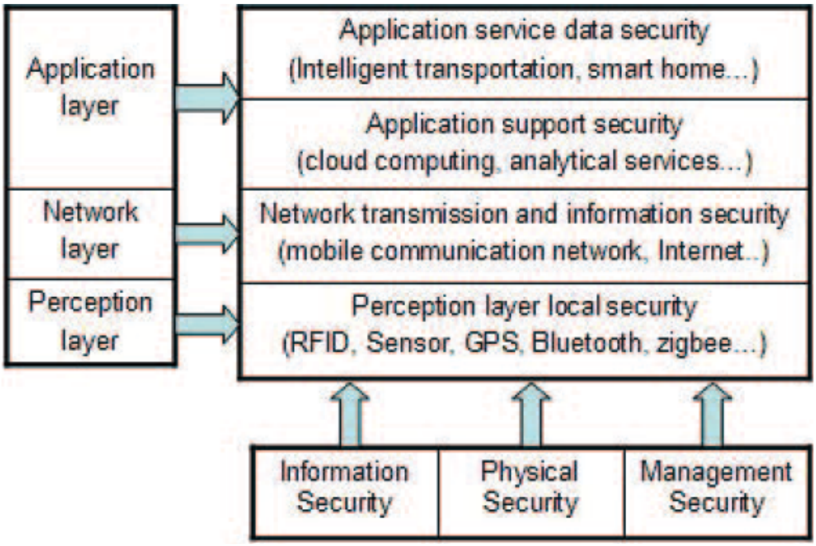

Application layer
Network layer
Perception layer
Application service data security (Intelligent transportation, smart home...)
Application support security (cloud computing, analytical services...)
Network transmission and information security (mobile communication network, Internet..)
Perception layer local security (RFID, Sensor, GPS, Bluetooth, zigbee...)
Information Security
Physical Security
Management Security

Fig. 2.1 Three layered structure of Internet of Things

Analyzing this scheme we can see that we divide the IoT Architecture in: Perception Layer, Network Layer and Application Layer, from bottom to top. The first one groups all the physical resources needed to gather data (sensors, RFID etc); the second one represents the classical network layer as intended in

ISO/OSI stack. The last one groups all the applications and services needed to the users to use the IoT infrastructure.

2.2 Security overview

As we can see from Figure 2.1, for every layer, we must ensure Information, Physical and Management Security. We can ensure these characteristics by focusing on 3 security concepts: Comprehensive Perception, Reliable Transmission and Intelligent Processing. The first one means that every sensor of the Perception Layer is used in a way that allows to gather data anytime and anywhere. Reliable Transmission means that the transmission of data must be pursued in the most efficient and secure way, through the wired or wireless network, to the data center, in real time. Intelligent Processing means that we can do middle-ware analysis to deal in the best way with collected information, to save resources before submitting data to the Application Layer.
In order to obtain the maximal level of security and privacy preservation, it's possible to use the above mentioned characteristics to point out the security problems in the IoT environment. The first three are traditional security features, the others are new ones [29].
1) The security problems of Perception layer data collection and

transmission: Sensor nodes have a high heterogeneity. They generally have simple structures and processors. These characteristics make the security protection complex.
2) The traditional security issues of Network Layer: Although Internet security architecture is very mature, there are still many kind of attacks. For example a large number of malicious nodes sending data at the same time will lead to a DoS attack. So the specific network should be built in order to fit transmission of IoT information
3) The Application Layer security problems: For different application fields, there are many complex and varied security issues.
4) The contradiction between Security and Costs: If each sensor node cost is too low, the large number of low performance nodes will reduce the overall security of the sensor network. On other hand, high performance nodes can increase security, but the cost of network maintenance will rise.
5) Lightweight: The processor performance of sensor nodes is limited, so there is a need for lightweight security authentication and encryption algorithms
6) Asymmetric: Compared with network terminals, gateway nodes data processing ability is weak, for this reason it's important to handle these asymmetric networks in the best way to be coordinated and to grant efficient security management measures.
7) Complexity: The type of applications determine the number

of security issues and their complexity. The security problem of each layer should be considered synthetically.
As mentioned above those problems are numerous and complex, therefore it's fundamental to understand all kinds of security issues of each layer, and potential attacks. Considered the system as a whole, the security problems should be solved at the beginning of the design. Therefore, literature [10] raises the application for the security state assessment about IoT based on grey correlation algorithm, which put several common attacks as the security factor, to realize quantitative evaluation of the entire network about environment and status. It also lists the specific steps to apply this algorithm to the security state evaluation. In the following sections there will be an evaluation of the security problems in each layer.

2.2.1 Perception Layer security problems

The main equipment in perception layer includes RFID, ZigBee, and all kind of sensors. When data are collected, the way to transmit information is wireless network transmission. The signals are usually public exposed, and if the network lacks of effective protection measures, they will be monitored, intercepted and disturbed easily. The most of sensing devices are deployed in unmanned monitoring sites, so the attackers can easily gain access to the equipment, controlling or physical tampering them.

For instance Differential Power Analysis (DPA) is a very effective attack.
Several kind of attacks are the following [15]:
1) Node Capture: Key nodes, such as gateway nodes, are easily controlled by the attackers. It may bring to information leaks, including group communication key, radio key, matching key etc, and then threats the security of the entire network.
2) Fake Node and Malicious Data: The attackers add a node to the system and input fake code or data through it. It will occasionally lead to stop the transmission of real data, but more important, the energy saving of mobile nodes is denied, leading to control or destruction of the entire network.
3) Denial of Service Attack: DoS attack is the most common attack in WSN and Internet. It causes loss of network resources and makes the service unavailable.
4) Timing Attack: By analyzing the time required for executing encryption algorithm the attacker can obtain key information.
5) Routing Threats: Through cheat, tamper or resend routing information, the attacker can create routing loops, cause or resist network transmission, extend or shorten the source path, form the error messages, increase the end-to-end delay, etc.
6) Replay Attack: Attacker sends a package which has been received by the destination host, in order to obtain the trust of the system. It's mainly used in the authentication process, destroying the validity of certification.
7) Side Channel Attack (SCA): The opponent attacks encryp-

tion devices through the information leakage in the side channel, such as time consumption, power consumption or electromagnetic radiations.
8) Mass Node Authentication Problem: The efficiency of mass node authentication needs to be solved in IoT. My thesis work is a proposal to solve this problem.
In addition, by the fact that mobile intelligent terminals will be an important part of IoT perception layer, their security can't be ignored. For example, most of smartphones have security bugs, combined with some of the features like phone memory scanning, address uploading and positioning, can lead attackers to threat the users' privacy. Therefore privacy protection is a key concept to address.

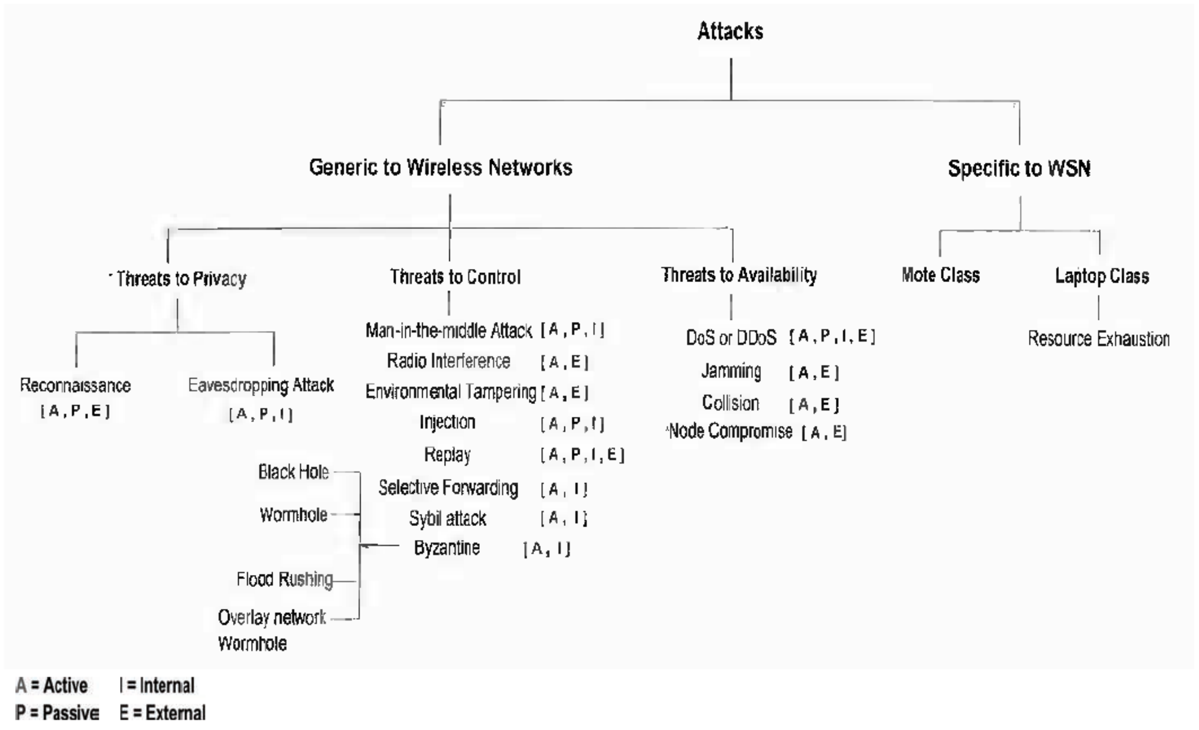


Fig. 2.2 Attack taxonomy for Wireless Sensor Networks [20]

Wireless Sensor Networks security attacks

Wireless sensor networks [20] are a subclass of wireless networks in general, so most kinds of attacks that can be directed at wireless networks can be directed at WSN. However, due to additional challenges, WSN breed a whole new set of attacks that can be classified into "mote-class attacks" and "laptop-class attacks". In a mote-class attack, the attacker takes control of a few compromised nodes and has capacity no greater than that of these ordinary sensor nodes. Hence, he has to launch attacks without depleting his resources (storage, computation, and bandwidth). Moreover, the attacker is restricted in his coverage due to limited transmission power. In contrast, the laptop-class attacker is resource-abundant (high transmitting power, longer battery life, high-speed processors, and highly receptive omnidirectional antennae). Hence, he has greater coverage and diversity in terms of the attacks that he can launch. Figure 2.2 portrays the attack taxonomy for WSN, while Table 2.1 shows the attack classification through a layered approach. In general, attacks can be divided into active and passive attacks.

- **Passive attack:** In this type of attack, the attacker is able to intercept and monitor data between communicating nodes, but does not tamper or modify packets for fear of raising suspicion of malicious activity among the nodes. For example, in traffic analysis, the attacker may not be able to decode encrypted data, but can find useful information by analyzing headers of packets, their sizes, and the frequency of transmission. In WSN, reconnaissance can also be performed to understand information exchange between communicating nodes, particularly at data aggregation points. Furthermore, routing information can be exploited using traffic analysis.

- **Active attack:** In this type of attack, the attacker actively participates in all forms of communication (control and data) and may modify, delete, reorder, and replay messages or even send spoofed illicit messages to nodes in the network. Some other active attacks include node capturing, tampering with routing information, and resource exhaustion attacks. Peculiar to WSN, the attacker can modify the environment surrounding sensors, which could affect the sensed phenomena.

2.2.2 Network Layer security problems

1) Traditional Security Problems: General security problems in communication networks will pose a threat to data confidentiality and integrity. Although the existing communication networks have relatively complete security protection measures, there are still some common threats, including illegal access to networks, eavesdropping information, confidentiality damage, integrity damage, DoS attacks, Man-in-the-Middle attacks,

Table 2.1 Attack classification using a layered approach

Layer	**Attack Vectors**
Application	Data corruption and repudiation
Transport	Session hijacking and SYN flooding
IP layer	Byzantine, resource consumption, location disclosure, wormhole and black hole
Data link	Traffic analysis
Physical	Interference and jamming
Multilayer attack	DoS, replay, man-in-the-middle and replay

virus invasion, exploit attacks, etc

2) Compatibility problems: The existing Internet network security architecture is designed on a person's perspective. This design doesn't necessarily apply to machine communication. Using the existing security mechanisms will lead to split the logic relationship between IoT machines. Therefore, by the fact that access networks have multiple access methods, the heterogeneity of these networks makes security, interoperability and coordination of networks becoming worse, leading to security vulnerabilities.

3) The Cluster Security Problems: Including network congestion, DoS attacks, authentication problems, etc. By the fact IoT is designed with the goal of using a huge number of devices, the uses of the existing modes of authentication for all of its devices will cause a large amount of data flowing through the network, likely causing its block. Moreover the existing IP technology does not apply to a large number of node identification. The mutual authentication among a lot of equipment causes serious waste of key resources.

4) Privacy Disclosure: With the development of the information retrieval technology and social engineering, hackers can easily collect a large number of particular users' privacy information.

2.2.3 Application Layer security problems

Depending on the application environment, the Application Layers are completely different, leading to various security issues. Nowadays there isn't an universal standard for the construction of IoT applications, some enterprises carry out M2M (machine-to-machine) mode IoT, such as intelligent community, intelligent household, medical, etc, which has some solutions developed on a 6LoWPAN (IPv6 over Low power WPAN) architecture, like medical sensory systems. Although application layer security is more complex and burdensome, it can still be summed up to some common security problems:

1) Data Access Permissions, Identity Authentication: Different applications have different users, likely a large amount of them. In order to prevent the illegal user intervention, the administrator should use effective authentication technology. Spam and

malicious information identification and processing should also be considered.
2) Data Protection and Recovery: Data communication involve user's privacy. Data protection mechanisms and data processing algorithms are not perfect, and they may cause data loss and even catastrophic damage. The mass node management is also one reason.
3) The ability of Dealing with Mass-Data: Due to the large number of nodes, we have to face the problem of huge amounts of data transmission in a very complex network environment. This can lead to network interruptions and data losses once the data processing ability can't meet the requirements.
4) The Application Layer Software Vulnerabilities: When writing software, because programmers write non-standard codes, occasionally buffer overflows vulnerabilities exist in the software, etc. Hackers can use those exploits to carry their purpose.

2.3 Security measures

As a Multi-layer fusion network, IoT security involves different layers in IoT. There are lots of security technologies applied in these independent networks, especially in the mobile communication network and the Internet network. For sensor networks in IoT, the diversity of the resources involved and the network heterogeneity make security research very complicated. This section will introduce the security measures for each layer.

2.3.1 Perception Layer security measures

Because RFID and WSN are a fundamental part of IoT Perception layer, their security measures will be introduced respectively.

1) RFID security measures [14]:

1. **Access Control:** It's used mainly in order to prevent the user's privacy leaks, to protect against free reading of the sensible information contained in the RFID tags, like label failure, chip protection, antenna energy analysis. The implementation of Access Control on RFID tags can be done easily in high-end tags, while on low-cost tags can be a little tricky. A good solution about Access Control on low-cost tags is presented in [12], which implements a Selective RFID Jamming.

    - *Passports [18]:* In the United States was introduced the idea of using RFID tags inside passports to contain personal information of citizens like document

ID, digital photos and biometric data (like the fingerprints or iris images). The protocol used in first implementation is BAC (Basic Access Control), which has been proved to be vulnerable to passive skimming and bruteforce, so a second protocol has been standardized: EAC (Extended Access Control) but yet not utilized.

- *Proximity Cards:* Used to grant access control to a determined class of workers inside a company. In this way the worker has just the privileges he needs to develop his job, and the required protection of his own personal data contained inside the tag (entry and exit time, transfers and so on) which could be used by an opponent, if not correctly protected. There are a lot of protocols which implements Access Control on RFID tags, one of them is illustrated in [4].

- *Automobile Ignition Keys:* In this case access control is used to bind a determined key to his own car, to avoid car theft through physical duplication of the key. Indeed the first authentication happens at the door opening, the tag transmit the ID code to the control unit, which enables the opening only if the ID code is correct. The same procedure happens at the car ignition. In this case the utilized protocols belong to the manufacturers.

- *Credit Cards:* Most updated bank circuits (Visa, Mastercard and American Express) introduced contactless version of their own credit cards. Access control in this case is required to not pass information just by standing near an RF reader. For this purpose containers like Faraday cage are used. Obviously protocols for Access Control are owned by manufacturers and kept secret.

2. **Data Encryption:** To grant data security of the RFID system, it's compulsory to encrypt the RFID signal using the appropriate algorithm. Literature [28] puts forward a kind of nonlinear key algorithm based on the displaced calculation, and achieves RFID system data encryption. To guarantee the condition of high speed data transmission, this key algorithm uses little computing power, still achieving very high security.

   - *Pharmaceutical Applications:* In the Pharmaceutical world, Data encryption is fundamental. Applications like drug tracing and e-pedigrees are used to recognize forged products, and obviously those data need to be protected to keep the supply chain protected from frauds. Another application which require strong data encryption is the Smart Labelling, indeed, smart labels contain patients health data which can be used to hurt their lives.

- *SpeedPass:* The SpeedPass is a contactless payment solution used in the United States to pay for example at gas stations or to stores which support this kind of payment. In this case Data Encryption is essential to protect bank data passing from SpeedPass to reader.
- *Smart Metering:* Any information about the electrical consume of a family, contains a lot of information about the family itself and the activity carried out inside an habitation. For this reason, it is compulsory to encrypt this kind of data.
- *Logistics:* Trough RFID sensors it's possible to keep track of goods, for example, inside a warehouse. For this reason is important to encrypt data regarding goods, to avoid an opponent to know if at a given time there will be the required quantity of items to be stolen.

3. **Security Channel based on IPSec:** IPSec protocol suite provides two types of security mechanisms: authentication and encryption. The receiver of IP communication data is able to confirm the sender's real identity trough authentication mechanism. Data encryption mechanisms, instead, prevent attackers to eavesdrop and tamper data during transmission, and encode data for ensure data confidentiality.

4. **Cryptography Technology Scheme:** Cryptography technology not only can realize the user privacy protection, it can also protect confidentiality, authenticity and integrity of RFID system. Secure communication protocols include random number generation, hash functions, server data search, and re-encryption mechanisms.

5. **Physical Security Scheme:** SCA is a major problem in physical security. The Differential Power Analysis is a common means of SCA. The various strategies adopted to prevent DPA can be divided into two categories: hiding and masking.The first one eliminates data dependencies of the energy consumption; masking makes the intermediate values of the encryption devices to be randomized in the process.

2) WSN security measures [20]:

As attacks on WSN become more sophisticated, the demand for new security solutions is continually increasing. Hence, an array of new security schemes have been designed and implemented in the past decade [9], [21]. Most of these schemes have been designed to provide solutions on a layer-by-layer basis rather than on a per-attack basis; in doing so, they have left a gap between layers that may lead to cross-layer attacks. In general, any security suite should ensure authentication, integrity, confidentiality, availability, access control, and nonrepudiation. In addition, physical safety is absolutely necessary to avoid tampering or destruction of nodes. Therefore, construction of tamper-resistant sensor nodes is absolutely necessary. However, such tamper-resistant schemes come at a higher manufacturing cost and are restricted to applications that are not only critical but that use fewer nodes.

- *Authentication:* The main objective of authentication is to prevent impersonation attacks. Hence, authentication can be defined as the process of assuring that the identity of the communicating entity is what it claims to

- *Integrity:* The goal of integrity is to affirm that the data received are not altered by an interceptor during communication (by insertion, deletion, or replay of data) and is exactly as it was sent by the authorized sender. Usually, cryptographic methods such as digital signatures and hash values are used to provide data integrity.

- *Confidentiality:* The goal of confidentiality is to protect the data from unauthorized disclosure. A common approach to achieve confidentiality is by encrypting user data.

- *Availability:* The goal of availability is to ensure that the system (network) resources are available and usable by an authorized entity, upon its request. It tries to achieve survivability of the network at all times.

- *Access control:* The goal of access control is to enforce access rights to all resources in its system. It tries to prevent unauthorized use of system and network resources. Access control is closely related to authentication attributes. It plays a major role in preventing leakage of information during a node compromise attack. One of the conventional approaches to access control is to use threshold cryptography. This approach hides data by splitting them into a number of shares. To retrieve the final data, each share should be received through an authenticated process.

- *Nonrepudiation:* Nonrepudiation can be best explained with an example. Let Alice and Bob be two nodes, who wish to communicate with each other. Let Alice send a message (M) to Bob. Later, Alice claims that she did not send any message to Bob. Hence, the question that arises is how Bob should be protected if Alice denies any involvement in any form of communication with Bob. Nonrepudiation aims to achieve protection against communicating entities that deny that they ever participated in any sort of communication with the victim.

1. **Security vs. Privacy Attacks**

    - *Encryption Algorithm:* They include both symmetric and asymmetric encryption Asymmetric ones use mainly RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curves Cryptography). Most common symmetric algorithms instead are Skipjack and RC5. By the fact that nodes processing ability is poor, symmetric algorithms are the most common to be used in WSN.

2. **Security vs. Control Attacks**

   - *Key Management:* The design of security requirements of WSN key management mainly reflects in the security of key generation or update algorithm, forward privacy, backward privacy and extensibility, against collusion attacks, source authentication and freshness. There are four main key distribution protocols: simple key distribution protocol, key predistribution agreement, dynamic key management protocol and hierarchical key management protocol.
   - *Authentication and Access Control:* Authentication techniques mainly include lightweight public key authentication technology, PSK (Pre Shared Key), random key pre-distribution authentication technology, auxiliary information authentication technology, one way hash functions authentication technology, etc. Access control mainly includes asymmetric cryptosystems and symmetric cryptosystems.

- *Intrusion Detection Technology:* IDS (Intrusion Detection System) can monitor the behavior of network nodes timely and find suspicious behavior of nodes.

3. **Security vs. Availability Attacks**

   - *Secure Routing Protocol:* An efficient security routing protocol algorithm generally uses the following mechanisms: clustering, data fusion, multiple hops routing mechanism, key mechanism, etc. SPINS security framework [22] is widely used in security routing technology, including SNEP (Secure Network Encryption Protocol) and μTESLA (Micro Timed Efficient Streaming Loss-tolerant Authentication protocol). SNEP is used to implement confidentiality, integrity, freshness and point-to-point authentication. μTESLA is an efficient flow authentication protocol based on time. It realizes point to multipoint broadcast authentication.
   - *Physical Security Design:* It mainly includes node and antennas design. Node design is developed by hardware structure and security chip selection, chip connection, radiofrequency circuit design, data acquisition unit design. Antenna design should meet good communication distance, high adaptability, stability, etc.

Security in WSN using a Layered Approach

1. **Security measures in Physical Layer:** To prevent radio interference or jamming, the two common techniques used are frequency-hopping spread spectrum (FHSS) and direct-sequence spread spectrum (DSSS). In FHSS, the signal is modulated at frequencies such that it hops from one frequency to another in a random fashion at a fixed time interval. The transmitter and the corresponding receiver hop between frequencies using the same pseudo-random code for modulation and demodulation. If an eavesdropper intercepts a FHSS signal, unless he has prior knowledge of the spreading signal code, he will not be able to demodulate the signal. Furthermore, spreading the signal across multiple frequencies will considerably reduce interference. In DSSS, a spreading code is used to map each data bit in the original signal to multiple bits in the transmitted signal. The pseudo-random code (spreading code) spreads the input data across a wider frequency range compared to the input frequency. In the frequency domain, the output signals appear as noise. Since the pseudo-random code provides a wide bandwidth to the input data, it allows the signal power to drop down below the noise threshold without losing any information. Therefore, this technique is hard for an eavesdropper to detect, due to lower energy levels per frequency and more

tolerance to interference. The above-mentioned schemes can provide security only as long as the hopping pattern or the spreading code is not disclosed to any adversary.

2. **Security measures in Data Link Layer:** Link-layer security plays an important role in providing hop-by-hop security. Its protocols are useful in handling fair channel access, neighbor-node discovery, and frame error control. To prevent denial-of-service (DoS) attacks on WSN, it is proposed that each intermediate node in the active routing path performs an authentication and integrity check. However, if a few intermediate nodes in the active path have very low energy levels, and if they are forced to perform authentication checks, they would expend all their energy and disrupt the active path. On the other hand, if we look at end-to-end authentication in WSN, it is more energy-efficient, since the sink node (resource-abundant) is the only node that performs authentication and integrity checks. Nevertheless, this scheme is vulnerable to many types of security attacks (black hole, selective forwarding, and eavesdropping). Hence there is a need for adaptive schemes that consider the energy levels of each node when deciding on the authentication schemes. Early security approaches focused on symmetric keying techniques, and authentication was achieved using Message Authentication Code (MAC). One of the common MAC schemes

is a cipher block chaining message authentication code. However, this scheme is not secure for variable-length input messages. Hence the end user (sensor nodes) have to pad the input messages to be equal to a multiple of the block cipher. Therefore, each node has to waste energy padding input data. To overcome this issue, other block cipher models such as CTR and OCB have been proposed. With reference to confidentiality, symmetric encryption schemes used to protect WSN are DES, AES, RC5, and Skipjack (block ciphers) and RC4 (a stream cipher). Usually, block ciphers are preferred over stream ciphers because they allow authentication and encryption. A few proposed link-layer security frameworks include TinySec, Sensec, SNEP, MiniSec, SecureSense [13], [22] and ZigBee Alliance (www.zigbee.org). However, these schemes have limitations. For example, in Tinysec a single key is manually programmed into all the sensor nodes in the network. A simple node-capture attack on any one of these nodes may result in the leakage of the secret key and compromising of the entire network. A need for a stronger keying mechanism is needed to secure TinySec. In addition, TinySec requires padding for input messages that are less than 8 bytes. It uses block cipher to encrypt messages, and for messages that are less than 8 by tes, the node will have to use extra energy to pad the message before encrypting.

WSN real life applications

There are a lot of real life applications for Wireless Sensor Networks in Internet of Things; here I'll point out some of them, but there is a whole universe of possible applications other than those listed here.

- **Smart Cities:**
  - Smart Parking
  - Smartphone Detection
  - Electromagnetic Fields Detection
  - Traffic Congestion
  - Smart Roads
  - Waste Management
  - Smart Lightning
  - Noise Urban Maps
  - Structural Health
- **Smart Environment:**
  - Forest Fire Detection
  - Air Pollution
  - Earthquake Early Detection
  - Landslide and Avalanche Detection

  - Snow Level Monitoring

- **Smart Water:**
  - Chemical Leakage Detection in rivers
  - Potable Water Monitoring
  - Pollution Levels in search
  - River Floods
  - Water Leakages
  - Swimming Pool Remote Measurement

- **Smart Metering:**
  - Smart Metering
  - Photovoltaic Installation
  - Silos Stock Calculation
  - Water Flow
  - Tank Level

2.3.2 Network Layer security measures

In the current structure of Internet of Things, Network Layer is the Internet or an existent communication network. There are several factors that endanger information safety on the Internet, and are the same factors which can pose a treat to the IoT information service. An important fact is that the Old network isnt' completely adapted to IoT. The traditional routing is simple and it's main goal isn't security. Because of IoT main characteristics like random node arrangement, autonomous infrastructure, unreliability of energy limitation and communication, they brings to not have a standard structure, but a dynamic topology and not fixed infrastructure. It will all lead to an attacker who can easily cause serious damages to the IoT system. For every different network architecture it's needed to set up the specific authentication cohesive mechanism, the end-to-end authentication and key agreement mechanism, PKI (Public Key Infrastructure), WPKI for wireless, Security routing, IDS, etc. Due to the huge amount of data, network availability have to be considered. In addition, it's also needed to strengthen cross-domain authentication in network layer. Also Network Virtualization is widely used, because it greatly reduces the complexity of network management and the possibility of wrong operations. With the development of the Next Generation Network (NGN), being a transport layer for IoT, the IPv6 technology has to be considered. Literature [25] gives the development trend of IPv6-based information security protocols. Also IPv6 networks security mechanisms and application of security products are discussed in detail.

2.3.3 Application Layer Security measures

Iot applications are very different from each other and it brings the Application Layer to be not uniformed. Indeed different applications have different security needs. For this reason the security approach to Application Layer is a bit different compared to the other 2 layers, and it's divided in two different solutions: Technical and Non-Technical.

Non-Technical Solutions

1. *Increasing the awareness of safety:* Let users realize the importance of information security and how to correctly use IoT services in order to reduce the confidential information leakage.

2. *Strengthen information security management:* It includes resource management, physical security information management, password management, etc.

Technical Solutions

1. *Network authentication across heterogeneous networks and key agreement:* It includes symmetric and asymmetric cryptosystems and certification transfer technology.

2. *Protection of private information:* It includes fingerprint technology, digital watermarking, anonymous authentication, homomorphic cryptography and threshold cryptography, which will be discussed later in this thesis work.

Chapter 3

Privacy protection cryptographic techniques

The Cryptographic techniques which we will examine in depth in this chapter are: Homomorphic Encryption, Anonymous Authentication and Threshold Cryptography.

3.1 Homomorphic Encryption

Homomorphic encryption is a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. This is sometimes a desirable feature in modern communication system architectures. Homomorphic encryption would allow the chaining together of

different services without exposing the data to each of those services, for example a chain of different services from different companies could calculate 1) the tax 2) the currency exchange rate 3) shipping, on a transaction without exposing the unencrypted data to each of those services. Homomorphic encryption schemes are malleable by design. This enables their use in cloud computing environment like IoT for ensuring the confidentiality of processed data. In addition the homomorphic property of various cryptosystems can be used to create many other secure systems, for example secure voting systems [23], collision-resistant hash functions, private information retrieval schemes, and many more. There are several partially homomorphic cryptosystems, and also a number of fully homomorphic cryptosystems. Although a cryptosystem which is unintentionally malleable can be subject to attacks on this basis, if treated carefully homomorphism can also be used to perform computations securely.
The difference between partially and fully homomorphic cryptosystems is that first one allows only some operations on ciphertexts (e.g., additions, multiplications, quadratic functions, etc.), while the latter allows arbitrary computation on ciphertexts.

3.1.1 Partially homomorphic cryptosystem

As stated before, partially homomorphic cryptosystems (PHE) allow only some operations on ciphertexts. Some of the encryp-

tion systems used nowadays are the following: *Unpadded RSA, ElGamal, Goldwasser-Micali, Benaloh, Paillier, Okamoto-Uchiyama, Naccache-Stern, Damgård-Jurik and Boneh-Goh-Nissin.*
Some example of IoT applications which can use PHE are:

1. *Protection of mobile agents:* Since all conventional architectures are based on binary strings, (and they only require addition and/or multiplication, PHE is perfect to protect this kind of agents. 2-way approach: Encrypted functions calculation or Encrypted data calculation.

2. *MultiParty Computation:* Will be described in section 3.1.2

3. *Threshold Schemes:* Will be described lately in section 3.3

4. *Zero-knowledge Proofs:* Used to prove the knowledge of some information, without revealing it.

5. *Watermarking and Fingerprint schemes:* Homomorphic property is used to apply a mark to a data earlier encrypted, in watermarking case. For fingerprint schemes, the user who buys a good has to be identifiable to be sure he doesn't redistribute data in illegal way. In an IoT environment it can be used like some sort of authentication to protect user's privacy.

6. *Mix-Nets:* Mix-Nets are protocols which provides server anonimation gathering encrypted messages from different users. A desirable property for this kind of protocols is re-encryption, achievable using homomorphic cryptography

7. *Data aggregation in WSN:* It's a technique which combine partial data obtained from single nodes, inside intermediate nodes, to reduce the communication overhead optimizing bandwidth usage of wireless links. In military and medical applications privacy of these data is critical, so the gathering has to be done keeping data protected also to intermediate nodes. For this reason homomorphic cryptography is used, to protect data and aggregate them when they are already encrypted through specific functions for WSNs.

3.1.2 Fully homomorphic cryptosystem

A cryptosystem that supports arbitrary computation on ciphertexts is known as fully homomorphic encryption (FHE) and is far more powerful. Such a scheme enables the construction of programs for any desirable functionality, which can be run on encrypted inputs to produce an encryption of the result. Since such a program never decrypts its inputs, it can be run by an untrusted party without revealing its inputs and internal state. The existence of an efficient and fully homomorphic cryptosystem would have great practical implications in the outsourcing of private computations, for instance, in the context of cloud computing. The utility of fully homomorphic encryption has been long recognized. The problem of constructing such a scheme was first proposed within a year of the development of RSA [24]. A solution proved more elusive; for more than 30 years, it was unclear whether fully homomorphic encryption was even possible. During that period, partial results included the Boneh–Goh–Nissim cryptosystem that supports evaluation of an unlimited number of addition operations but at most one multiplication, [2] and the Ishai-Paskin cryptosystem that supports evaluation of (polynomial-size) Branching program [11].

Gentry FHE scheme

Craig Gentry [7], using lattice-based cryptography, described the first plausible construction for a fully homomorphic encryption scheme. Gentry's scheme supports both addition and multiplication operations on ciphertexts, from which it is possible to construct circuits for performing arbitrary computation.The construction starts from a somewhat homomorphic encryption scheme, which is limited to evaluating low-degree polynomials over encrypted data. (It is limited because each ciphertext is noisy in some sense, and this noise grows as one adds and multiplies ciphertexts, until ultimately the noise makes the resulting ciphertext indecipherable.) Gentry then shows how to slightly modify this scheme to make it bootstrappable, i.e., capable of evaluating its own decryption circuit and then at least one more operation. Finally, he shows that any bootstrappable somewhat homomorphic encryption scheme can be converted into a fully homomorphic encryption through a recursive self-embedding. For Gentry's "noisy" scheme, the bootstrapping procedure effectively "refreshes" the ciphertext by applying to it the decryption procedure homomorphically, thereby obtaining a new ciphertext that encrypts the same value as before but has lower noise. By "refreshing" the ciphertext periodically whenever the noise grows too large, it is possible to compute arbitrary number of additions and multiplications without increasing the noise too much. Gentry based the security of his scheme on the assumed hardness of two problems: certain worst-case problems over ideal lattices, and the sparse (or low-weight) subset sum problem. Gentry's Ph.D. thesis [8] provides additional details. Regarding performance, ciphertexts in Gentry's scheme remain compact insofar as their lengths do not depend at all on the complexity of the function that is evaluated over the encrypted data, but the scheme is impractical, and its ciphertext size and computation time increase sharply as one increases the security level. The main problem of Gentry's scheme was that it's completely impractical, because it exponentially growths the calculation time to grant a security level comparable to the other similar algorithms.

Bruce Schneier said that: "Gentry's scheme is completely im-

practical... Gentry estimates that performing a Google search with encrypted keywords - a perfectly reasonable simple application of this algorithm - would increase the amount of computing time by about a trillion." After the publication of Gentry's phd thesis IBM has created a tool called HELib to implement FHE.
In 2011 Gentry, Brakerski and Vaikuntanathan [3]proposed a new FHE scheme which drastically improve Gentry scheme and sets up security on weaker hypothesis, all without bootstrapping procedure. This proposal sets up on Learning-Without-Error or Ring-LWE problems, which have a security level of 2^n against known attacks. The latter offer equal results of the first with worst performances. Results are:

- Gentry per-gate computation: $O(n^{3.5})$
- RLWE L-level arithmetic circuits per-gate computation: $O(nL^3)$
- RLWE L-level arithmetic circuits per-gate computation with optimization: $O(n^2)$

On-The-Fly MultiParty Computation (MPC)

Another FHE application scheme came out in 2012 from Lopez-Alt, Tromer and Vaikuntanathan [16], who proposed a new vision of secure multiparty computation using FHE. In this proposal was indeed possible for the cloud to non-interactively perform arbitrary, dynamically chosen computations on data belonging to arbitrary sets of users chosen on-the-fly. All user's input data and intermediate results are protected from snooping by the cloud as well as other users. This extends the standard notion of fully homomorphic encryption (FHE), where users can only enlist the cloud's help in evaluating functions on their own encrypted data. In on-the-fly MPC, each user is involved only when initially uploading his (encrypted) data to the cloud, and in a final output decryption phase when outputs are revealed; the complexity of both is independent of the function being computed and the total number of users in the system. When users upload their data, they don't need to decide in advance which function will be computed, nor whom they will compute with; they need only retroactively to approve the eventually chosen functions and on whose data the functions were evaluated.

3.2 Anonymous Authentication

Anonymous Authentication allows a server to confirm a particular user's privileges to use a chosen service authenticating in the system without exposing hiw own identity. It allows to be protected against attackers who want to track and identify online users.

3.2.1 Anonymous Authentication in IoT

In 2012 Alcaide et al. [1] proposed a form of Anonymous Authentication usable in an IoT environment. The roles used in the algorithm are two: *Users* and *Data Collectors*. First ones generate data and can authenticate themselves anonymously and unlinkably on Data Collectors (DC) possessing an Anonymous Authentication Certificate (AAC). DCs are the entities responsible for gathering data transmitted from authorized users. It implies that DCs have to verify that users transmitting data have correct AACs. Main characteristics of this protocols are:

- It doesn't set up on a central unit, neither for setup phase, implying that all the parameters necessary for the protocol to work are generated in a cooperative way by all system nodes.

- Users generate and distribute a private key through (t,n) Threshold Cryptography 3.3 allowing the system to have until t users compromised without the system being compromised itself.

- This protocol is based on the scheme proposed by Persiano and Visconti in 2004.

3.3 Threshold Cryptography

A threshold cryptosystem [5] consists of the four following components:

- A *key generation algorithm* takes as input a security parameter k, the number l of decryption servers, the threshold parameter t, and a random string w; it outputs a public key PK, a list $SK_1, \ldots, SK_l$ of private keys and a list $VK, VK_1, \ldots, VK_l$ of verification keys.

- An *encryption algorithm* takes as input the public key PK, a random string w and a plaintext M; it outputs a ciphertext c.

- A *share decryption algorithm* takes as input the public key PK, an index $1 \leq i \leq l$, the private key SK_i and a ciphertext c; it outputs a decryption share c_i and a proof of its validity $proof_i$.

- A *combining algorithm* takes as input the public key PK, a cyphertext c, a list $c_1, \ldots, c_l$ of decryption shares, the list $VK, VK_1, \ldots, VK_l$ of verification keys and a list $proof_1$,...,$proof_l$ of validity proofs; it outputs a plaintext M or fails.

3.3.1 Application Scenario

To analyze security properties we simulate a “game” including the following players: a dealer, a combiner, a set of l servers P_i, an adversary and users. All are considered as probabilistic polynomial time Turing machines. We consider the following scenario:

1. In an initialization phase, the dealer uses the key generation algorithm to create the public, private and verification keys. The public key PK and all the verification keys VK, VK_i are publicized and each server receives its share SK_i of the secret key SK.

2. To encrypt a message, any user can run the encryption algorithm using the public key PK.

3. To decrypt a ciphertext c, the combiner first forwards c to the servers. Using their secret keys SK_i and their verification keys VK, VK_i each server runs the decryption algorithm and outputs a partial decryption c_i with a proof of validity of the partial decryption $proof_i$. Finally the combiner uses the combining algorithm to recover the cleartext if enough partial decryptions are valid.

3.3.2 Security requirements

We consider an adversary able to corrupt up to t servers. Such a corruption can be passive, i.e. the attacker only eavesdrop the servers. It can also consist in making the servers fail and stop. Finally, it can be active; in this last case, the adversary completely controls the behavior of the corrupted servers. In the following, we only consider non-adaptive adversaries who choose which servers they want to corrupt before key generation.
A threshold cryptosystem is said to be *t-robust* if the combiner is able to correctly decrypt any ciphertext, even in the presence of an adversary who actively corrupts up to t-servers.
All messages are sent in clear between each server and the combiner. Moreover, the combining algorithm which takes each partial decryption and recovers the cleartext is public and can be executed by any server as they see all decryption parts. So the only assumption we make about the communication channel is the existence of a broadcast channel between all participants.

Threshold semantic security

All the encryption schemes we study are semantically secure. Informally speaking, let us consider an attacker who first issues two messages M_0 and M_1; we randomly choose one of these messages, we encrypt it and we send this ciphertext to the attacker. Finally, she answer which message has been encrypted. We say that the encryption scheme is semantically secure if there

exists no such polynomial time attacker able to guess which of the two messages has been encrypted with a non-negligible advantage.
We extend the definition of semantic security to threshold cryptosystems in the setting where an attacker who actively, but non-adaptively, corrupts t servers learns not only the public parameters, as in the regular cryptosystem, but also the secret keys of the corrupted servers, the public verification keys, all the decryption shares and the proof of validity of those shares.
Let us consider the following game A:

- **A1:** The attacker chooses to corrupt t servers. She learns all their secret information and she actively controls their behavior.
- **A2:** The key generation algorithm is run; the public keys are publicized, each server receives its secret key and the attacker learns the secrets of the corrupted players.
- **A3:** The attacker chooses a message M and a *partial decryption oracle* gives her l valid decryption shares of the encryption of M, along with proofs of validity. This step is repeated as many times as the attacker wishes.
- **A4:** The attacker issues two messages M_0 and M_1 and sends them to an *encryption oracle* who randomly chooses a bit b and sends back an encryption c of M_b to the attacker.

- **A5:** The attacker repeats step A3, asking for decryption shares of encryptions of chosen messages.

- **A6:** The attacker outputs a bit b'

A threshold encryption scheme is said to be semantically secure against active non-adaptive adversaries if for any polynomial time attacker, $b = b'$ with probability only negligible greater than $1/2$.

Notice that our definition of semantic security reduces to the original one when we consider only one server $(l = 1)$ who knows the secret key, and an adversary who does not corrupt any server $(t = 0)$. In this case, steps A3 and A5 just consists into encrypting chosen plaintexts and this can be done without the help of *partial decryption oracle*.
Finally the previous game may not be confused with the chosen ciphertext attack security described by Gennaro and Shoup [6]. The attacker can only ask for partial decryptions of ciphertext for which she already knows the corresponding plaintext. The goal of steps A3 and A5 is to prove that partial decryptions give no information about the private keys of the non-corrupted servers. Since the cryptosystems we study are not immunized against chosen ciphertext attacks in the non-distributed case, we cannot expect to extend such a property to threshold versions.

Security Proofs

The aim is to provide robust threshold version of semantically secure cryptosystems. Our security proofs are based on reduction; we prove that if an adversary can break the semantic security of the threshold cryptosystem, then she must be able to break the semantic security of the initial cryptosystem.
We show how to build an adversary to attack the semantic security of the traditional cryptosystem from an adversary who can break the security of the threshold cryptosystem. The basic idea in order to use an attacker against the threshold version and to turn her into an attacker against the traditional cryptosystem, is to simulate all the extra information that are not provided in a traditional attack.
More precisely, if we are able to simulate the public verification keys and the secret keys of corrupted servers in step A2, the decryption shares and their proof of validity in steps A3 and A5 in such a way that the adversary cannot distinguish between the real distribution and the simulated one, we can feed this adversary with simulated data in order to obtain an attacker against the semantic security of the initial non-distributed scheme. Consequently, the security proofs consist into showing how to simulate all those data.

3.3.3 Shamir threshold secret sharing scheme

In 1979, Shamir proposed a protocol to share a secret element s of a field F between l servers in such a way that any group of $t+1$ servers can efficiently recover s but any coalition of t servers is not able to gain any information about the secret.
The scheme is based on the Lagrange interpolation formula that allows to compute $P(X)$, for any $X \in F$, if P is a polynomial of degree t and if $t+1$ values of $P(x_i)$ are known for $t+1$ distinct elements $x_1, \dots, x_{t+1}$ of F:

$$P(X) = \sum_{i=1}^{t+1} \prod_{j=1;j\neq 1}^{t+1} \frac{X-x_j}{x_i-x_j} \times P(x_i)$$

In order to share a secret a, a polynomial P of degree t is randomly chosen in $F[X]$ such that $P(0)=s$ and each server receives a point $(x_i, P(x_i))$ with $x_i \neq 0$. The Lagrange formula shows that the knowledge of $t+1$ such points allows to recover the whole polynomial P and consequently $P(0)=s$. One can also prove that knowledge of t points gives no information about s.

3.3.4 Paillier cryptosystem

Various cryptosystems based on randomized encryption schemes $E(M)$ which encrypt a message M by raising a basis g to the power M have been proposed so far. Their security is based on the intractability of computing discrete logarithm in the basis g without a secret data, the secret key, and easy using this trapdoor. We call those cryptosystems *trapdoor discrete logarithm schemes*. As an important consequence of this encryption technique, those schemes have homomorphic properties that can be informally stated as follows:

$$E(M1+M2) = E(M1) \times E(M2) \text{ and } E(k \times M) = E(M)^k$$

Paillier has presented three closely related such cryptosystems in [19]. We only remind the first one. This cryptosystem is based on the properties of the Carmichael lambda function ($\lambda(\bullet)$)in $\mathbb{Z}^*_{n^2}$. We recall here the main two properties:
For any $w \in \mathbb{Z}^*_{n^2}$,

$$w^{\lambda(n)} = 1 mod\, n, \text{ and } w^{n\lambda(n)} = 1 mod\, n^2$$

Key Generation

Let n be an RSA modulus $n = pq$, where p and q are prime integers. Let g be an integer of order n modulo n^2. The public key is $PK = (n, g)$ and the secret key is $SK = \lambda(n)$.

Encryption

To encrypt a message $M \in \mathbb{Z}_n$, randomly choose x in $\mathbb{Z}^*$ and compute the ciphertext $c = g^M x^n mod\, n^2$.

Decryption

To decrypt c, compute $M = \frac{L(c^{\lambda(n)} mod\, n^2)}{L(g^{\lambda(n)} mod\, n^2} mod\, n)$ where the L function takes in input elements from the set $\mathbb{S}_n = \{u < n^2 | u = 1 mod\, n\}$ and $L(u) = \frac{u-1}{n}$.

The integers $L(c^{\lambda(n)} mod\, n^2)$ and $L(g^{\lambda(n)} mod\, n^2)$ are equal to 1 when they are raised to the power n so they are the n^{th} roots of unity. Furthermore, such roots are of the form $(1+n)^{\beta} = 1 + \beta n\, mod\, n^2$. Consequently, the L-function allows to compute such values $\beta\, mod\, n$ and
$L((g^M)^{\lambda(n)} mod\, n^2) = M \times L(g^{\lambda(n)} mod\, n^2) mod\, n.$

Security

It is conjectured that the so-called composite residuosity class problem, that exactly consists in inverting the cryptosystem, is intractable. The semantic security is based on the difficulty to distinguish n^{th} residues modulo n^2. We refer to [19] for details.

3.3.5 Paillier threshold cryptosystem

3.3.6 Key generation algorithm

Choose an integer n, product of two strong primes p and q, such that $p = 2p' + 1$ and $q = 2q' + 1$ and $\gcd(n, \varphi(n)) = 1$. Set

$m = p'q'$. Let β be an element randomly chosen in $\mathbb{Z}_n*$. Then randomly choose $(a,b) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^*$ and set $g = (1+n)^a \times b^n$ mod n^2. The secret key SK=$\beta \times m$ is shared with the Shamir scheme: let $a_0 = \beta m$, randomly choose t values a_i in $\{0,\dots,n \times m - 1\}$ and set $f(X) = \sum_{i=0}^{t} a_i X^i$. The share s_i of the i^{th} server P_i is $f(i)$ mod nm. The public key PK consists of g,n and the value $\theta = L(g^{mb}) = am\beta$ mod n. Let VK= V be a square that generates the cyclic group of squares in $\mathbb{Z}_{n^2}^*$. The verification keys VK_i are obtained with the formula $v^{\Delta s_i}$ mod n, recalling that $\Delta = l!$ with l number of nodes in the group.

Encryption algorithm

To encrypt a message M, randomly pick $x \in \mathbb{Z}_n^*$ and compute $c = g^M x^n$ mod n^2.

Share decryption algorithm

The i^{th} player P_i computes the decryption share $c_i = c^{2\Delta s_i}$ mod n^2 using his secret share s_i. He makes a proof of correct decryption which assures $c^{4\Delta}$ mod n^2 and v^{Δ} mod n^2 have been raised to the same power s_i in order to obtain c_i^2 and v_i.

Combining algorithm

If less than t decryption shares have valid proofs of correctness, the algorithm fails. Otherwise, let S be a set of t+1 valid shares and compute the plaintext

$$M = L(\prod_{j \in S} c_j^{2\mu_{0,j}^S} \bmod n^2) \times \frac{1}{4\Delta^2\theta} \bmod n$$

where $\mu_{0,j}^S = \Delta \times \prod_{j' \in S \setminus \{j\}} \frac{j'}{j'-j} \in \mathbb{Z}$

Security evaluation

We proved that Paillier threshold cryptosystem is secure against an active, non adaptive adversary and this gives the indistinguishability (IND) characteristic to the algorithm. As mentioned before, the homomorphic property of the Paillier threshold cryptosystem makes this system malleable, but, introducing the random number r in the encryption procedure, avoid the possibility to create a meaningful malevolent message m'. By the above characteristics, this scheme is defined as IND-CCA2 (indistinguishability under adaptive chosen cyphertext attack) which is the necessary property for a Public Key cryptosystem.

Chapter 4

Group Authentication in IoT

4.1 Application scenario

The purpose of this scheme is to give MANETs nodes 4.1the chance to authenticate themselves in a lightweight and scalable way. It verifies the authenticity of each node which wants to participate in group activity. Possible scenarios in which this scheme can be used are mainly military scenarios (where high level of security is required) and disaster areas (where high security is suggested). The proposed scheme allows multi-to-multi authentication in group applications. An example of possible use of this scheme is to create a sort of an encrypted VPN where all members can safely talk, being sure no inside or outside attackers can disrupt their communication. A possible application is to create hierarchy groups to let Generals talk to Generals, Lieutenant-Generals with Lieutenant-Generals and Generals, and so on. In summary to allow a hierarchical and secure way of communication.

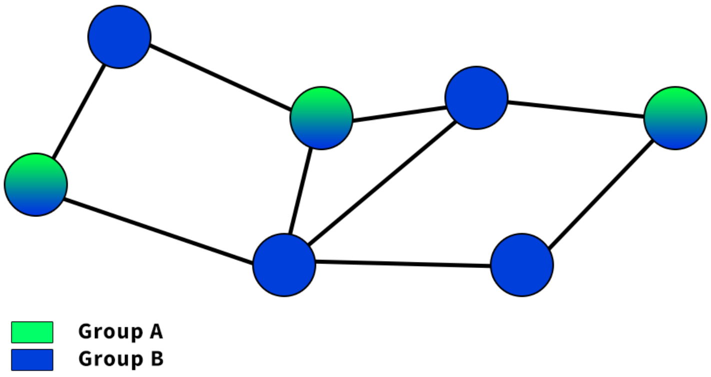


Fig. 4.1 Manet structure: blue and green nodes belong to both group A and B, while blue nodes belong only to group B

The proposed scheme was born to face the typical problem of constrained resources in an IoT environment. In particular, we use it in a Mobile Ad-hoc NETwork scenario which accomplish the above mentioned characteristic. To decentralize the computational effort of the proposed scheme, let all members of the group to authenticate each other, in a Multi-to-Multi way, avoiding a single point of failure in the system, improving the overall security of this scheme.
To achieve this goal we used Paillier Threshold Cryptography, which is a public key variant of the (t,n) threshold scheme where t is threshold and n is number of group members. This cryptog-

raphy scheme helps to achieve homomorphic properties which is helpful in preserving privacy of members but also dangerous, because it makes this scheme *Malleable* (we will explain after the security issues). By the fact it's a probabilistic asymmetric public key encryption system, which uses randomness in the encryption algorithm, encrypting the same plaintext several times produces a different cyphertext each time. The proposed scheme is used to verify the authenticity of all the members taking part in a group oriented application, it also establish a shared secret key between the members of the group which can be used for further communication in any group oriented application.

4.2 Mobile Ad-hoc NETwork

Future information technology [27] will be mainly based on wireless technology. Traditional cellular and mobile networks are still, in some sense, limited by their need for infrastructure (i.e., base stations, routers). For mobile ad hoc networks, this final limitation is eliminated. Ad hoc networks are key to the evolution of wireless networks. They are typically composed of equal nodes that communicate over wireless links without any central control. Although military tactical communication is still considered the primary application for ad hoc networks, commercial interest in this type of networks continues to grow.

Applications such as rescue missions in times of natural disasters, law enforcement operations, commercial and educational use, and sensor networks are just a few possible commercial examples. Ad hoc wireless networks inherit the traditional problems of wireless and mobile communications, such as bandwidth optimization, power control, and transmission quality enhancement. In addition, the multihop nature and the lack of fixed infrastructure generates new research problems such as configuration advertising, discovery, and maintenance, as well as ad hoc addressing and self-routing. In mobile ad hoc networks, topology is highly dynamic and random. In addition, the distribution of nodes and, eventually, their capability of self-organizing play an important role. The main characteristics can be summarized as follows:

- The topology is highly dynamic and frequent changes in the topology may be hard to predict.
- Mobile ad hoc networks are based on wireless links, which will continue to have a significantly lower capacity than their wired counterparts.
- Physical security is limited due to the wireless transmission.
- Mobile ad hoc networks are affected by higher loss rates, and can experience higher delays and jitter than fixed networks due to the wireless transmission.

- Mobile ad hoc network nodes rely on batteries or other exhaustible power supplies for their energy. As a consequence, energy savings are an important system design criterion. Furthermore, nodes have to be power-aware: the set of functions offered by a node depends on its available power (CPU, memory, etc.).

A well-designed architecture for mobile ad hoc networks involves all networking layers, ranging from the physical to the application layer.
Despite the fact that the management of the physical layer is of fundamental importance, there has been very little research in this area: nodes in mobile ad hoc networks are confronted with a number of problems, which, in existing mobile networks, are solved by the base stations. The solution space ranges from hierarchical cell structures (a self-organized pendant of cellular networks) to completely ad hoc, stochastic allocations. Power management is of paramount importance. General strategies for saving power need to be addressed, as well as adaptation to the specifics of nodes of general channel and source coding methods, radio resource management, and multiple access. Mobile ad hoc networks do not rely on one single technology; instead, they should be able to capitalize on technology advances. One challenge is to define a set of abstractions that can be used by the upper layers and still not preclude the use of new physical layer methods as they emerge. Primitives of such an abstraction are,

for example, the capabilities and covering ranges of multicast and unicast channels. Information such as node distribution, network density, link failures, etc., must be shared among layers, and the MAC layer and the network layer need to collaborate in order to have a better view of the network topology and to optimize the number of messages in the network. Mobile ad hoc networks have the unique characteristic of being totally independent from any authority or infrastructure, providing great potential for the users. In fact, roughly speaking, two or more users can become a mobile ad hoc network simply by being close enough to meet the radio constraints, without any external intervention. Moreover, telecommunication networks are expected to grow with the advent of new (and totally unexpected) applications. Although in the past telecommunication networks were studied and developed as separate building blocks, for users of mobile ad hoc networks the interaction between higher layers and lower layers is essential. Resilient and adaptive applications that can continue to perform effectively under degraded conditions can significantly enhance network operations from a user's perspective. Such applications can also significantly ease the design pressure in complex engineering areas such as quality of service (QoS) and mobile routing at the network layer [17]. Communication among layers is the only practical approach to a demanding environment that raises issues that rarely occur in other networks [26].

4.3 Proposed scheme

The proposed scheme is composed by three parts, a **Phase 0**, in which every node creates a couple (PR_i, PU_i) used for securing the communications, a **Pre-phase**, in which the group is formed, and an **Authentication phase**, where the users belonging to the group are authenticated and a secret session key is shared in order to secure further group communications. Every step is carried out by particular and predefined messages: *Join, Req, Resp* and *Leave*, which will be described later in this chapter.

4.3.1 Phase 0

In this phase every node belonging to the MANET creates a couple (PR_i, PU_i) in order to secure further communications, and broadcast the Public key to all the MANET nodes.

4.3.2 Pre-phase

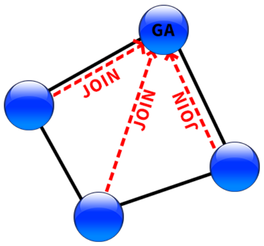
In this phase, every node who knows the group credentials asks a Group Authority (which can be any ordinary node of the MANET, usually the one which starts the group activity, GA from now on) to join the group sending a *Join* message (Figure 4.2(a)) to it in a secure way. This is made by means of the couple (PR_i, PU_i) generated and broadcasted in phase 0 by simply encrypting the Join message with the GA Public Key. When GA has checked the correctness of the credentials, it updates the parameters n

and t, it creates a new couple of keys, PR_G and PU_G, and then it divide the PR_G into n shares by Shamir secret sharing (as explained in the subsection 3.3.6). At this point the GA sends all the shares to the nodes, encrypting them with the recipient Public Key, (Figure 4.2(b)) securing that no inside or outside attackers can eavesdrop the share.

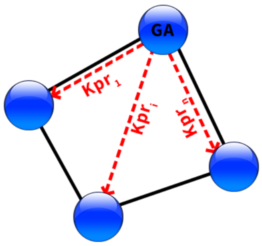
4.3.3 Authentication phase

In order to start group activity, it has to be performed the group authentication, as a pre-requisite to check if all the members $M_1, \ldots, M_m$ (where $t \leq m \leq n$) are part of the group. The GA chooses a pseudo-random number as a session secret $[SS]$ key which is going to be shared with all the members of the group once the group authentication is done. This is encrypted using the public key PU_G of the group, and sent to all the active members of the group (Figure 4.2(c)). It is possible to recognize active from passive members by a procedure performed by GA periodically.

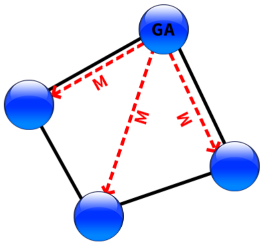
When a communication session needs to be started, and periodically after that, GA sends a *Req* message to all the group members (Figure 4.3(a)). They can actually answer with a *Resp* message or no respond at all (Figure 4.3(b)). Every time a Group Activity Check procedure is performed, GA updates its list of active members and the number m. It has to be performed every x minutes, where $x < SS_{refreshtime}$ and if m doesn't change SS is



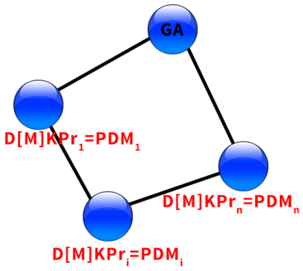
(a) Join request



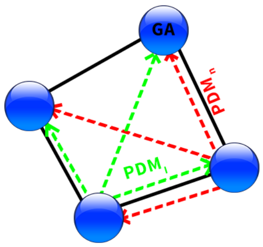
(b) Share distribution



(c) Message broadcasting



(d) PDMs creation



(e) PDMs broadcasting

Fig. 4.2 Authentication phase

not changed.

If a member doesn't respond after a predefined timeout, it's assumed as not active. After that, GA starts the **Authentication phase** and each of the members, upon receiving the message

$$M = \{[SS]_{PU_G}, H[SS], N1\}$$

applies his private key part to decrypt it (Figure 4.2(d)), obtaining in this way his own Partial Decryption Message (PDM from now on). Each user has his own unique PDM, corresponding to a different share of the group's Private Key PR_G.

$$PDM_i = Decrypt(M)_{Kpr_i}$$

Each device then sends his PDM to every active member of the group, including GA (Figure 4.2(e)). All the devices wait to combine all the PDMs until $m-1$ are received. The advantage of the threshold comes up in this step, indeed with only one initial distribution of the shares K_{pr_i} (Figure 4.2(b)) the group can start communications with a number $t \leq m \leq n$ of members, and m is always known thanks to the Group Activity Check procedure.
Each device so combine all the m received PDMs in order to obtain the decrypted session key.

$$SS = Combine(PDM_1, PDM_2, \ldots, PDM_m)$$

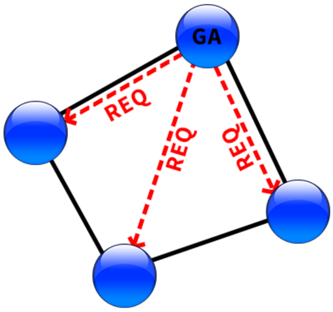
If combining all PDM shares is successful, the session key is obtained and it proves that all the members are part of that group. The decrypted session key is hashed to get $H'[SS]$ and checked with the one sent by the GA. If $H'[SS] = H[SS]$, then authentication is successful, and session key $[SS]$ is obtained. It can be used for further communications between the devices, and it can be carried out by using Symmetric Key Encryption. If combining all PDM shares is unsuccessful, it means that there is at least one non-member in the group, and, hence, the activity cannot be initiated, and group authentication fails. Further individual authentication needs to be performed to identify the non-member in the group.

4.4 Messages exchange

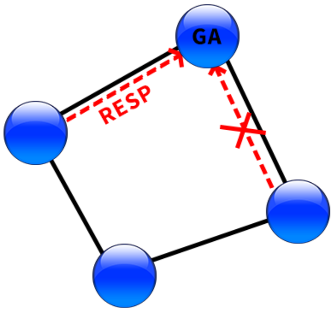
Possible messages, as said before, are four: *Join, Req, Resp, Leave*.

4.4.1 Join

Join message is used by a node to join the Group (Figure 4.2(a)); it contains the Group credentials, the node ID, an encrypted hash, a timestamp (to avoid replay attacks, a nonce is not enough because GA doesn't respond to this message, so the node can't



(a) Req broadcasting



(b) User responses

Fig. 4.3 Group Activity Check

check the nonce, and the timestamp is not replayable by an attacker because message is authenticated by an encrypted hash) and all is encrypted with the GA public key.

$$Join = \{TYPE = 1|(G_id, G_pw)|node\ ID|H[join - H]_{PR_i}|TS\}_{PU_{GA}}$$

When a node sends the Join message, it starts a timeout, based on physical distance between MANET nodes; if it doesn't receive his share until timeout is expired, it sends the Join message again, changing Timestamp and the Hash.

When GA receive the Join message, it updates the values (t,n) and start the **Pre-phase**.

4.4.2 Req

Req message is used either to start a group activity or to update the GA's list of active nodes (Figure 4.3(a)). The node which starts the group activity is elected as GA for the current session. Req message is sent to all *n* nodes which have a share.

$$Req = \{TYPE = 2|GA\ ID|H[Req - H]_{PR_{GA}}|Nonce\}_{PU_{dst}}$$

When GA sends the Req message, it starts a timeout, based on physical distance between MANET nodes, if it doesn't receive responses from each of n nodes until the timeout is expired, it updates the list of active nodes and handles the non-responding nodes as not active. If a node is not active for x times consecutively, GA update (t, n) and begin the **Pre-phase**.

When a node receive the Req message, it can respond with either a *Resp* message or no respond at all.

4.4.3 Resp

Resp message is sent by nodes to GA as a response to a *Req* message, to communicate their presence in the upcoming communication session.

$$Resp = \{TYPE = 3|node\ ID|Nonce + 1\}_{PU_{GA}}$$

When a node sends the Resp message, it just waits to receive the *M* message (Figure 4.2(c)).

When GA receives the Resp message, it updates the active member list, and waits either to receive all n messages or the expiring of Timeout (Figure 4.3(b)). Then it sends the M message to the *m* active nodes.

4.4.4 Leave

Leave message is sent by nodes to GA in order to leave the group.

$$Leave = \{TYPE = 4|node\ ID|H[Leave - H]_{PR_i}\}_{PU_{GA}}$$

When a node sends the Leave message it will not be able to communicate with the group until he'll do another Join.

When GA receives a Leave message, it updates the values (t,n) and begins a new **Pre-phase** with updated values.

4.5 Implementation

The implementation of this work has been done on a Manet composed by 5 nodes. Each of them is an HP Compaq tc4200 with following specifications:

- **O.S.:** Linux Ubuntu 10.04 with 2.6.38 kernel
- **CPU:** Intel Celeron M370 (1.50 Ghz, 1MB L2 Cache, 400MHz FSB)
- **RAM:** 2GB DDR2
- **Wireless card:** Intel PRO/Wireless 2200BG (802.11b/g)

By the fact that the wireless card has very poor features, we used a wireless usb card, a TP-Link TL-WN722N, with following specifications:

- 802.11b/g

- 150Mbps
- Ad-Hoc and Infrastructured modalities.

The characteristics of those PCs are chosen to simulate in the best way constrained resources to fit better IoT environment characteristics. Every node has his own static IP from the range 10.0.0.3 to 10.0.0.7. The programming language used is Java™ SE Runtime Environment (build 1.8.0_40 – b25) and the only pre-compiled library used is the *Paillier Threshold Encryption Toolbox* created by Dallas Texas University, which uses particular maths to lighten cpu's calculation effort. The code is composed by a Deamon running on every node which first reads a configuration file in which are present the initial conditions, and then runs silently until it's waken up by the beginning of some phase. To let the nodes receive, send and process data at the same time, it has to be built up a Multithreaded Server-Client running on every node.

4.5.1 Multithreaded Server

```
 Thread t = new Thread(new ServerHandler(
myPR,pv,sv,pdmc),"Receiving Server");

 t.start();
```

where ServerHandler() is the class which implements the runnable used to receive and synchronize the classes and the objects involved in the code. The other parameters are *myPR* which is a PaillierPrivateKey Object containing the Private Key of the node; pv, sv and pdmc are classes used to synchronize the various objects through different threads. Since we used a multithreaded server-client architecture, it's needed a way to let the different threads alert each other when a shared object is ready to be used by other threads. For this reason every shared class is synchronized and has 2 particular methods: *get*() and *put*(). The first one is used by a thread to request an object contained in one of the following classes: pv, sv and pdmc, containing a share, an encrypted SS and a PDM respectively. In more detail, the way used to let a thread communicate another that an object is ready is the following: *get*() contains a *wait*() method, which puts the thread who called it in a sleeping phase. The thread which is processing the object, when is ready, calls a *put*(), which contains a *notify*() method, used to wake up the sleeping thread. In this way there are no chances from the thread calling *get*() to work on an object not ready to be processed. So, the shared classes pv, sv and pdmc passed to the class Server-Handler to let all the threads work on the same, synchronized, objects. Here is an example of the *get*() and *put*() methods:

```
public synchronized void put(BigInteger SS){

        this.SS=SS;
        token=true; //PPTK is usable
        notify();
        }
```

```
public synchronized BigInteger get(){

        if(!token)
```

```
                    try{
                            wait();
                    }catch(InterruptedException exc){
                            exc.printStackTrace();
                    }

            return SS;
}
```

The ServerHandler in turn, contains in his *run*() method, the method used to receive objects, which is *newClientHandler(clientSocket, PrivKey, pv,sv,pdmc)* As stated before, to use the same objects in different threads, we passed *PrivKey* (the private key of the node), the other structures listed before and the Object clientSocket, used to open a receiving socket whenever a node wants to transmit some objects to my node. Here are parts of the code used in the receiving part, contained in the newClientHandler class.

```
ObjectOutputStream oos = new ObjectOutputStream(
clientSocket.getOutputStream());
ObjectInputStream ois = new ObjectInputStream(
clientSocket.getInputStream());
BigInteger[] msg =(BigInteger[]) ois.readObject();
```

Here is declared and associated an Object Stream to the clientSocket opened in the ServerHandler class, and then is casted and put in an array of BigInteger the message received. It's put in this form because the encryption and decryption is operated on BigIntegers, and into array to allow flexible key dimension. At this point some processing is elaborated on the first element of the array, which contains encrypted data useful to understand what message we just received, and then a switch case is inserted:

- *Case 1:* Received a Share
- *Case 2:* Received the encrypted SS
- *Case 3:* Received the PDM from GA
- *Case 4 to n+3:* Received the correspondent node PDM

Cases 4 to n+3 are very important (and serializable in the code) for the pdmc structure, which puts one of the received PDMs in its position, and only when at least *m* PDMs are arrived, its method $put()$ gives the $notify()$ to the main thread to let him proceed with the combining algorithm. To transmit, it's used a method *ObjectTransmit(port, ipv4, msg)* which takes as input the port and the ipv4 address used to communicate with the recipient, and the array of BigIntegers msg. Here's the code:

```
clientSock = new Socket(ipv4,port);
System.out.println("Connecting...");
ObjectOutputStream oos = new
ObjectOutputStream(clientSock.getOutputStream());
ObjectInputStream ois = new
ObjectInputStream(clientSock.getInputStream());
msg = (String) ois.readObject();
```

```
if (msg.equals(rx condition )){
    System.out.println("OK, proceding to send Msg");
    oos.writeObject(Bmsg);
```

4.5.2 Encryption and Decryption

The encryption and decryption side of the code is mainly created by the *Paillier Threshold Encryption Toolbox*, using PaillierPrivateKey, PaillierKey and Paillier classes (and some others..). Each one of them is a class containing different fields to fit the structure of Paillier Cryptosystem. To encrypt an object (a number or a string) it has to be put in a BigInteger, we use them because longs are not capacious enough to contain all the bits we use for a key. Here's an example of how to encrypt a number or a string. To encrypt a String object it has to be converted in a BigInteger.

```
byte[] bytemsg=myString.getBytes();
BigInteger M=new BigInteger(bytemsg);

Paillier esys=new Paillier();
esys.setEncryption(myPR);
BigInteger Cyph=esys.encrypt(M);
```

Depending on how long has to be the Public or Private key (it's possible to choose inside the script the number of digits the key has to be long, from 32 to 256) we have to work on chunks of dimension lesser than key's one. So I worked with Electronic CodeBook mode (ECB) trunking BigIntegers into parts of the right dimension to encrypt or decrypt.

4.5.3 Daemons

Every Daemon but the GA's one, is the same, just loading the configuration file it sets up the number of initial nodes, their IPs and the group credentials (which in a real application are pre-distributed to every node). Differently the GA node has his own Daemon in my implementation, however the possibility to have only one daemon for every node is practicable, just setting the GA's block of code in a *if* case, with the information of which node has to be the GA contained in the cfg file. The initial informations contained in the cfg file are:

1. The max number of nodes n contained in the group.
2. The dimension k of the Group's Public (and consequently Private) key.
3. The threshold law to calculate t.
4. The group credentials.

Every Daemon so reads the cfg file and sets itself up. Now we see an example of the logical operations the GA has to do, and then we'll see the single node's ones.

```
1)loads the cfg file

PHASE 0

1)Generate it's couple PR, PK
int nodeID=nID;
Random random = new Random();
long seed = random.nextLong();
PaillierPrivateKey pr= KeyGen.PaillierKey(s, seed);
PaillierKey pu= pr.getPublicKey();

2)Put Public Key into a file
to be broadcasted

String pubkey ="src/files/PublicKey"+nodeID;
try {FileWriter File= new FileWriter(pubkey);
        PrintWriter out=new PrintWriter(File);
        out.println("n:" + pu.getN());
        out.close();}
        catch(IOException e){
                System.out.println(e);
        }
```

```
PRE-PHASE

2)generates the shares

int l=n;
int w= (l/2)+1;
Random rnd=new Random();
```

```
long ls = rnd.nextLong();
String Strkeys ="chiavi";
KeyGen.PaillierThresholdKey(Strkeys, s, l, w, ls);
String[] chiavi = new String[l];
for (int i=0; i<l; i++){
        chiavi[i] = "chiave" + (i+1);
                }
PaillierPrivateThresholdKey[] keys =
KeyGen.PaillierThresholdKeyLoad(Strkeys);
for(int i = 0; i < keys.length; i++) {
        FileWriter File= new FileWriter(chiavi[i]);
        PrintWriter out=new PrintWriter(File);
        out.println("l:" + l);
        out.println("w:" + w);
        out.println("v:" + keys[0].getV());
        out.println("n:" + keys[0].getN());
        out.println("combineSharesConstant:"
+ keys[0].getCombineSharesConstant());
        out.println("s" + ":" +
        keys[i].getSi().toString());
        out.println("v" + ":" +
        keys[i].getVi()[i].toString());
        out.close();
}

3)split the shares in k-1 size
4)encrypt shares with dst keys
5)transmit the encrypted shares
```

```
AUTHENTICATION PHASE
```

```
1)Generation of SS

BigInteger[] SS = new BigInteger[1];
SS=new BigInteger(int numBits, Random rnd);

2)Encrypt SS with PU of GA

BigInteger encryptedSessionSecret =
AbstractPaillier.encrypt(SS[0],r,
TKeys[1].getThresholdKey());

3)Split eSS into pieces to fit
key dimension

4)Transmission of encrypted SS (eSS)
to various nodes

MTMultiClient.objectTransmit(
8080,IPofA,eSStoTx);

5)Calculation of my PDM

PartialDecryption myPDM=new PartialDecry-
ption(TKeys[0],encryptedSessionSecret);

6)Split and encrypt PDM with
dst node Public KeyGen

BigInteger[] ePDMBItoP=new Big-
Integer[num_of_pdm_cnk];
esys.setEncryption(NodePRs[1].getPu-
blicKey());
for (int i=0;i<num_of_pdm_cnk;i++){
        ePDMBItoP[i]=esys.encr-
ypt(PDMBI[i]);
}

7)Transmit PDM

MTMultiClient.objectTran-
smit(8080,IPofP ,ePDMBItoP);

8)Put my PDM in right position
into the rx structure

pdmc.putWithPosition(myPDM, 1);

9)Wait to have all PDMs rdy
PD=pdmc.get();

10)Combine PDMs

PaillierThreshold Message = new Pail-
lierThreshold(TKeys[0].getThresholdKey());
Message.setDecryptEncrypt(TKeys[0]);
SS=Message.combineShares(PD)
```

At this point we can see how the Daemon is constructed for peripheral nodes. The only difference is that they don't do the processing to create SS and the shares, but only wait to receive them, in order first Shares, then SS, so they calculate their PDM and send it to other nodes, waiting with pdmc.get() to receive all the PDM. Here's the code

```
1)Starting receiving server

Thread t = new Thread(new ServerHand-
ler(myPR,pv,sv,pdmc),"Receiving Server");
t.start();

2)Waiting his share
PPTK=pv.get();

3)Waiting eSS
SS=sv.get();

4)Calculate PDM
PartialDecryption myPDM=new Partia-
lDecryption(PPTK,SS);

5)Send PDM
6)Wait other PDMs
7)Combine
```

After all nodes received PDMs and correctly recover SS they can use symmetrical encryption with any kind of algorithm working on the desired length of the generated key.

4.6 Results

In this section we will first evaluate a theoretical time analisys and then we'll show some of the results obtained from the simulations on the above mentioned framework.

4.6.1 Time analisys

For the first phase, the *Phase 0*, the time analysis is quite simple, indeed every node have to broadcast his own public key to the other $n-1$ nodes. It brings the time complexity to be $O(n)$ for every node, considering the whole system it will raise to $O(n^2)$.
The second phase, the *Pre-Phase*, has same complexity of the first part of the Phase 0, so time complexity of Pre-Phase is $O(n)$.
Third phase is similar to Phase 0, so time complexity is $O(n)$ for the distribution of SS and $O(n^2)$ for the whole system to transmit and receive PDMs from the other nodes. This is actually a good result for a distributed system, allowing the infrastructure to grow to quite good dimensions with the authentication process scaling with it in a polynomial time.

4.6.2 Implementation results

In the implementation we misured the time that our nodes used to process the data and to transmit them around the MANET. First row of table 4.1 represent the time that 3 nodes need to

Table 4.1 Processing and Transmission time

	Average
3 nodes preprocessing	315,760 ms
5 nodes preprocessing	367,931 ms
SS distribution	185,633 ms

have the shares ready to be used, second row shows the same parameter but for an implementation with 5 nodes. The third row instead represent the time it's needed for the algorithm to converge, starting when GA creating the Session Secret, ending when all the nodes have their own copy of decrypted SS. All of the times in this table are measured through the $System.nanotime()$ java command, which gives the result in nanoseconds, and it is rounded to milliseconds to avoid system inaccuracy. Those results are measured with a distance from nodes of about 5 meters one from each other.

4.7 Security evaluations

As mentioned in section 3.3.6, we can assume that this algorithm is IND-CCA2 secure, which is the required security level needed for an asymmetric cryptosystem, it implies that our system is protected against *Outside Attackers* (opponents not belonging to the MANET system). For *Inside Attackers* instead (Attackers belonging to the MANET but not to the group), protection comes from the threshold cryptosystem, in particular mode, this algorithm gives protection against chosen/adaptive cyphertext attacks, and most critical and common ones to a MANET environment: Replay and Man-in-the-middle attacks. Protection against first one is well explained in section 4.4.1, a carefull system of nonce and timestamps avoid the consistency of Replay attacks. For Mitm attacks instead, protection is granted by the asymmetric encryption of every message exchanged in the authentication process.

Chapter 5

Conclusions

At the end of this thesis we can conclude that security is a fundamental concept nowadays, therefore it is important to realize that money and resources spent in information and privacy protection is not a loss, but an investment instead. The application realized in this thesis has multiple utilization in different environments, not only for MANETs, by the fact it is designed to face the main IoT concerns. One of the key concepts on which I based my work, is that the algorithm used in my application will work in the most scalable and lightweight way possible, allowing hundreds of devices to authenticate themselves without causing service denial. This has been realized distributing equally the calculation effort between system nodes. Moreover, this application makes possible the authentication of all the nodes participating to the group for the whole duration of the communication process, not only at the beginning. This is realized thanks to the Group Activity Check procedure; the application also distributes a symmetric key to encrypt further communications, which is simpler than using asymmetric keys encryption, with almost no security loss. In the end I would like to conclude this thesis with a reminder from Bruce Schneier, security responsible for IBM:

"If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology."

References

[1] Almudena Alcaide, Esther Palomar, José Montero-Castillo, and Arturo Ribagorda. Anonymous authentication for privacy-preserving iot target-driven applications. *Computers and Security*, 37:111–123, 2013. URL http://dblp.uni-trier.de/db/journals/compsec/compsec37.html#AlcaidePMR13.

[2] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of the Second International Conference on Theory of Cryptography*, TCC'05, pages 325–341, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-24573-1, 978-3-540-24573-5. doi: 10.1007/978-3-540-30576-7_18. URL http://dx.doi.org/10.1007/978-3-540-30576-7_18.

[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1115-1. doi: 10.1145/2090236.2090262. URL http://doi.acm.org/10.1145/2090236.2090262.

[4] Yu-Yi Chen, Meng-Lin Tsai, and Jinn ke Jan. The design of rfid access control protocol using the strategy

of indefinite-index and challenge-response. *Computer Communications*, 34(3):250–256, 2011. URL http://dblp.uni-trier.de/db/journals/comcom/comcom34.html#ChenTJ11.

[5] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In Yair Frankel, editor, *Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 90–104. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42700-1. doi: 10.1007/3-540-45472-1_7.

[6] Rosario Gennaro, Tal Rabin, Stanislaw Jarecki, and Hugo Krawczyk. Robust and efficient sharing of RSA functions. *J. Cryptology*, 20(3):393, 2007. doi: 10.1007/s00145-007-0201-2. URL http://dx.doi.org/10.1007/s00145-007-0201-2.

[7] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-506-2. doi: 10.1145/1536414.1536440. URL http://doi.acm.org/10.1145/1536414.1536440.

[8] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[9] M. Healy, T. Newe, and E. Lewis. Security for wireless sensor networks: A review. In *Sensors Applications Symposium, 2009. SAS 2009. IEEE*, pages 80–85, Feb 2009. doi: 10.1109/SAS.2009.4801782.

[10] Gao Hongbo. Study of the application for the security state assestment about the internet of things based on grey cor-

relation algorithm. *J.Manufactoring Automation. 34(11)*, 2012.

[11] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594. Springer, 2007. ISBN 3-540-70935-5. URL http://dblp.uni-trier.de/db/conf/tcc/tcc2007.html#IshaiP07.

[12] Ari Juels, Ronald L. Rivest, and Michael Szydlo. The blocker tag: Selective blocking of rfid tags for consumer privacy. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 103–111, New York, NY, USA, 2003. ACM. ISBN 1-58113-738-9. doi: 10.1145/948109.948126. URL http://doi.acm.org/10.1145/948109.948126.

[13] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *in Proc of the 2nd Int'l Conf on Embedded Networked Sensor Systems*, pages 162–175, 2003.

[14] Chen Li. System security solutions of rfid system of internet of things sensing layer. *J. Net Security technologies and Applicatin, (6)*, pages 34–36, 2011.

[15] Zhang Li. Principle and application of wireless sensor network. *M.Beijing: China Machine Press*, 2008.

[16] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1219–1234, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1245-5.

doi: 10.1145/2213977.2214086. URL http://doi.acm.org/10.1145/2213977.2214086.

[17] J.P. Macker, V.D. Park, and M.S. Corson. Mobile and wireless internet services: putting the pieces together. *Communications Magazine, IEEE*, 39(6):148–155, Jun 2001. ISSN 0163-6804. doi: 10.1109/35.925683.

[18] Mônica Nogueira and Noel Greis. *Uses of RFID Technology in US Identification Documents*. Institute for Homeland Security Solutions, 2009.

[19] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag. ISBN 3-540-65889-0. URL http://dl.acm.org/citation.cfm?id=1756123.1756146.

[20] Harsh Kupwade Patil and Stephen A. Szygenda. *Security for Wireless Sensor Networks Using Identity-Based Cryptography*. Auerbach Publications, Boston, MA, USA, 1st edition, 2012. ISBN 1439869014, 9781439869017.

[21] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: Security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, September 2002. ISSN 1022-0038. doi: 10.1023/A:1016598314198. URL http://dx.doi.org/10.1023/A:1016598314198.

[22] Adrian Perrig, Robert Szewczyk, JD Tygar, Victor Wen, and David E Culler. Spins: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[23] Lecturer Ron Rivest and Scribe Ledlie/ortiz/-paskalev/zhao. Lecture notes 15: Voting, homomorphic encryption, 2002.

[24] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[25] Zhang Shen. Development trend of ipv6-based information security products in network layer of iot. *27th National Computer Securit Academic Communication. (8)*, 2012.

[26] Martha Steenstrup, editor. *Routing in Communications Networks*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1995. ISBN 0-13-010752-2.

[27] I. Stojmenovic. *Handbook of Wireless Networks and Mobile Computing*. Wiley Series on Parallel and Distributed Computing. Wiley, 2003. ISBN 9780471462989. URL http://books.google.it/books?id=V5HFbsHgn_wC.

[28] Wei Wang. cipher algorithm in data transmission of rfid system on the internet for things. *J. Juornal of Beijing Information Science and Technology University*, 2009.

[29] Kai Zhao and Lina Ge. A survey on the internet of things security. In *Computational Intelligence and Security (CIS), 2013 9th International Conference on*, pages 663–667, Dec 2013. doi: 10.1109/CIS.2013.145.

Acknowledgements

Certamente non è facile ringraziare tutte le persone che hanno lasciato un segno, piu o meno importante, nel mio percorso universitario. Vorrei cominciare con il mio primo relatore, il Professor Stefano Giordano, che mi ha portato ad apprezzare il mondo delle reti, e che mi ha permesso di arrivare a questo punto della mia carriera fiero del percorso scelto. Quindi vorrei ringraziare il Professor Michele Pagano e l'Ingegnere neopapà Christian Callegari, che mi hanno permesso di lavorare a questa tesi, e che mi hanno supportato in tutti i modi possibili, dal primo all'ultimo giorno di lavoro. Uno speciale ringraziamento va al mio amico Giuseppe, senza il quale parte di questa tesi probabilmente sarebbe ancora in alto mare. Quindi vorrei ringraziare i miei amici, universitari e non, che mi hanno accompagnato in questo percorso e che lo hanno reso più leggero ed affrontabile, in particolare un sentito ringraziamento ad Andrea ed Alberto, che mi hanno dato una grossa mano a migliorare e ad affrontare gli impegni con più determinazione. Tra i ringraziamenti non può ovviamente mancare il gruppo di ragazzi sparsi per tutta Italia con cui ho trascorso serate memorabili a dir poco, in particolare Nicola e Raul, che hanno contribuito non poco al mio percorso di formazione personale. Una dedica molto speciale va necessariamente alla mia ragazza Elena, che mi ha aiutato a sorpassare ostacoli apparentemente insormontabili, rendendomi l'uomo che sono adesso. Ma soprattutto mi sento di ringraziare con tutto me stesso la mia famiglia, che mi ha permesso con innumerevoli sacrifici di arrivare a concludere il mio ciclo di studi, spero di avervi resi fieri quanto io lo sono di voi. A voi dedico questo mio conseguimento.