Advances in Computer Science and Technology

Chuan-Kun Wu

Internet of Things Security

Architectures and Security Measures

Springer

Advances in Computer Science and Technology

In cooperation with the China Computer Federation (CCF)

The Advances in Computer Science and Technology series publishes state-of-the-art results contributed by China Computer Federation (CCF) members or authors associated with or invited by CCF. This series aims to efficiently disseminate advanced researches and practical applications to international research community. The topical scope of Advances in Computer Science and Technology spans the entire spectrum of computer science and information technology ranging from foundational topics in the theory of computing to information and communications science and technology and a broad variety of interdisciplinary application fields. This series mainly include monographs, professional books and graduate textbooks, edited volumes and books. All volumes are authored or edited by established experts in their fields, published according to rigorous peer review, based on the editors' preview and selection and adequate refereeing by independent experts. Each volume will provide a reasonable self-contained account of a topic, as well as a survey of the literature on the topic. The intended audience includes graduate students, researchers, professionals, and industrial practitioners.

More information about this series at http://www.springer.com/series/13197

Chuan-Kun Wu

Internet of Things Security

Architectures and Security Measures

Springer

Chuan-Kun Wu
School of Information Science
and Engineering
Linyi University
Linyi, Shandong, China

ISSN 2198-2686 ISSN 2198-2694 (electronic)
Advances in Computer Science and Technology
ISBN 978-981-16-1371-5 ISBN 978-981-16-1372-2 (eBook)
https://doi.org/10.1007/978-981-16-1372-2

Preface

The concept of Internet of Things (IoT) has been proposed for decades. It is designed to connect small physical devices and RFID tags together over the networks, particularly over the Internet. The networked things can be physical things without computation capabilities, or devices that have computation and communication capabilities. Things without computation capabilities need to attach RFID tags so that they have digital identities and can be networked.

In an IoT system, application data are collected by sensors, transmitted over public and private networks, and processed in a data processing center, which may be used by end users. Command data from end users are transmitted down to the endpoint devices, including RFID tags.

The process of data collection makes an effective connection between the physical world and digital data, while the process of endpoint device control is also a connection between the digital data and the physical world, so the techniques of IoT are beyond the edge of the the Internet is only about digital data, and the IoT can connect the physical world with the digital data. If the Internet together with data and related network services is treated as a cyberspace, then the techniques of IoT provide a connection between the physical world and cyberspace. This small but substantial characteristic makes IoT substantially different from the Internet in the traditional sense.

There is no formal definition about what IoT is; however, the concept of IoT can be described by an IoT architecture. A number of different IoT architectures have been proposed in public literatures, including three-layer architectures, four-layer architectures, five-layer architectures, and six-layer architectures or domain-based architectures. Each architecture is a specific description about the logical structure of IoT. Although different architectures have some differences, the natural components of IoT remain the same. Therefore, different IoT architectures can all be referred to in the construction of IoT systems. Hence, it is not important to scrutinize the concepts of IoT and the descriptions of the IoT architectures. As long as a description of the IoT concept or IoT architecture is self-contained, it would not affect practical constructions and applications of IoT.

The techniques of IoT can be used widely. In recent years, the techniques of IoT have already been used in many areas, including industry, government, infrastructure, transport, facilities, and services, in everyday life.

Like many other techniques, while IoT brings many conveniences and advances, it also brings new security threats. Currently, security techniques for IoT systems are far from satisfactory. With the advances of IoT applications, problems caused by lack of security protection appear to be more severe. There have already been security attacks on IoT devices, which caused severe security events where large amount of IoT devices were involved in a DDoS attack. Given that the number of IoT devices keeps growing, DDoS attacks making use of victim IoT devices can be a big security threat and challenge in IoT security.

The current embracing situation with IoT security is that IoT has some specific features that are substantially different from the traditional Internet. Such features restrict the traditional security techniques to be smoothly planted into IoT. The feature of resource constraint of many IoT devices gives rise to the problem of lightweight security techniques, including lightweight encryption algorithms, lightweight authentication protocols, lightweight key management schemes, and lightweight implementations of cryptographic algorithms. The feature of connection between physical devices and digital data requires IoT to have operation security (OT security), in additional to the information security (IT security) as required by many traditional information systems. This monograph describes some differences between IT security and OT security and some specific techniques mechanisms for OT security.

It is noticed that the IoT security is targeted at practical applications; hence, a general description of security requirements is not enough. It is known that the primary model of information security covers confidentiality, integrity, and availability, known as the CIA triad, which is too rough for practical implementation, thus the IoT security should be more specified with detail. This monograph provides comprehensive security measures; each security measure has a brief description about the test methods. The listed security measures are meant to provide a closer and more practical viewpoint of how IoT security techniques can be practically equipped.

The IoT is still a new area irrespective of its concept having been proposed for decades. The IoT security is hence a new subject. This monograph is an attempt to systematically describe what the security techniques of IoT systems are about, including an architectural picture of IoT security, specific security requirements, security measures, and security test methods. The viewpoint and technical description of IoT security are based on the author's immature understanding and some immature research results, so they may not be convincing. This monograph provides some material for further discussion, with the possibility of promoting the advances of IoT security, both in theoretical research and practical applications.

The contents of this monograph are described as follows. Chapter 1 gives a general introduction about the concept of IoT, the components of IoT, some typical applications of IoT, and security issues of IoT. It is pointed out that the security protections are essential for IoT systems to work properly, because IoT systems

normally face many network attacks. Chapter 2 presents a number of different IoT architectures, including layered architecture and a domain-based architecture. This chapter describes the relations among those architectures and concludes that those different architectures are essentially the same; differences are noticed because the IoT systems are viewed from different angles. Chapter 3 presents an IoT security architecture, which is based on a 4-layered IoT architecture (i.e., perception layer, network layer, processing layer, and application layer). Apart from the layered components, there are also two supporting pillars in the IoT security architecture. Security requirements in each of the components of the IoT security architecture are briefly described. Chapter 4 introduces some fundamental knowledge of cryptography, including symmetric key ciphers, public key cryptosystems, hash functions, message authentication codes, identity authentication protocols, and the lightweight feature of cryptographic algorithms. Effort has been put to make the description less technical, because detailed technical description can be found in other cryptographic books. This chapter is mainly about the concepts of the techniques, and the about mechanism of how they work. Chapter 5 introduces the trust mechanism and key management problems in IoT. Trust in the network environment is the core basis for identity authentication and key management. Chapter 6 presents some security requirements and techniques for IoT perception layer security. Chapter 7 presents some security requirements and techniques for IoT network layer security. Chapter 8 presents some security requirements and techniques for IoT precessing layer security. Chapter 9 presents the privacy protection problem in IoT applications, which is a specific kind of IoT application security. Chapter 10 presents the security problems of RFID security, which is also a specific kind of IoT application security. Chapter 11 discusses the relations between information security (IT security) and operation security (OT security), addresses how OT security is different from IT security, and claims that IoT security should cover both IT security and OT security. Chapter 12 presents a comprehensive set of security measures, mostly focusing on the perception layer of IoT. These security measures are more specified and detailed description about how security services should be implemented in practice.

Although each chapter has a focus, the whole picture of IoT security is better described by the whole monograph. In fact, the contents in Chap. 4 are fundamental knowledge in cryptography, and the contents in Chap. 7 are common network security protocols; more detailed technical description can be found in other network security books. The inclusion of those contents is to make this monograph self-contained and to give people in the IoT domain a rough picture of fundamental cryptography and network security protocols, thus helping readers better understand IoT security as a whole.

The techniques and theories of IoT security is far from being mature, and some of the contents in this book only represent the viewpoint of the author, which may not be correct. Due to the knowledge limitation of the author, the time restriction in writing this monograph, and the status of IoT security techniques being immature, there are inevitably grammatical, logical, or even technical mistakes in this monograph. The unexpected COVID-19 pandemic more or less affected the writing process.

Finally, the author would like to express sincere thanks to the editors at Springer, specifically Dr. Lanlan Chang and Ms. Kripa Guruprasad, for their regular reminders and encouragement, particularly during a long period of self-isolation due to the COVID-19 pandemic, and their careful revision of the early versions of this monograph. Their support helped to make this monograph completed in time.

The author would also like to appreciate the support by Shandong Provincial Key Research and Development Program (Major Science and Technological Innovation Project, No.2019JZZY010134).

Linyi, China Chuan-Kun Wu
December 30th, 2020

Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter introduces some basic concepts related to Internet of Things (IoT), including specific IoT applications and their characteristics. This chapter points out some special characteristics of IoT security, including the lightweight feature of information security, some special features of operation security, which is related to illegal control over endpoint devices. The special features of IoT security indicate that traditional security techniques are often inappropriate and insufficient for IoT systems, particularly when the IoT systems have resource-constrained devices, or they are concerned with operation security such as in industrial applications.

1.1 The Concept of Internet of Things

The Internet of Things (IoT) refers to uniquely identifiable objects and their virtual representations in an Internet-like structure. The term Internet of Things was proposed by Kevin Ashton in 1999 when he tried to solve the problem of efficiently find one shade of lipsticks which were sold out quickly and hence became not easy to find, even if there were such lipsticks in store. The concept of the Internet of Things became popular through the Auto-ID Center at MIT which was set up by Kevin Ashton.

Radio-frequency identification (RFID) is an important component of the Internet of Things. In the Auto-ID Center at MIT, RFID was seen to be a prerequisite for the Internet of Things. However, RFID simply provides a way of identification, essentially it does not have difference from other kinds of coded identities.

With the development and wide applications of IoT, the concept of IoT becomes broader: an IoT can be based on sensor networks, where endpoint sensor devices collect environmental data and transmit the data to a data processing center for further processing and application.

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_1

In 2005, the International Telecommunication Union (ITU) published its annual report [23], named as The Internet of Things. It covered topics including new enabling technologies, business opportunities, public policy challenges, and implications for the developing world.

Under the support of European Commission (EC), the Cluster of European Research Projects on the Internet of Things (CERP-IOT) published a report in September 2009: *Internet of Things Strategic Research Roadmap*, pointing out that the Internet of Things connect physical and virtual things/devices with an unique identifier based on standard and co-operational communication protocols. Those devices collect environmental data of the physical world and would give reactions when being instructed or triggered. The networked things can be involved in commercial, social, and virtual activities by interaction over the networks. Users' application devices can query the status of the networked things, in this process security and privacy issues need to be properly managed [9].

The wide applications of IoT will gradually cover more and more industries, infrastructure surveillance, and living services in everyday life, such as smart grids, smart home, intelligent/wise medical systems, smart logistics, and smart transport. The importance of IoT has been beyond the attention of industry, it has attracted attention from national development and national strategy plan organizations.

The concept of Internet of Thing has been evolving since its birth, from RFID-based Internet of Things [13], to sensor-based ones [49]; from finding things on the Internet, to information collection and command execution over networks; from information handling to physical device control. Now the Internet of Things can be treated as a means for connection between the virtual world consisting of information and data (i.e., the cyberspace) to the physical world.

The Internet of Things is a system with multiple cutting-edge technologies, including nanotechnology, new material, microelectronics, networking and communications, and intelligent computing. The development of Internet of Things has created new concepts and corresponding technologies, including cloud computing, fog computing, edge computing, and has also encountered new security challenges, where the challenges come from the resource constraint of IoT devices, and the interaction between the information system (in the cyberspace) and real devices (in the physical world).

Today the communication network is well developed, capable of offering communication and networking services for billions of people. Mobile communications have wide bandwidth and low cost, hence are able to provide audio/video/multimedia services, and are able to support services of Internet of Things. However, the security services provided by the mobile networks as well as other kind of traditional networks are insufficient to meet security requirements of many IoT applications.

The Internet of Things has been used in more and more areas and industries. Although some traditional technologies for communications and networks can be tailored to the Internet of Things, there will be some specific techniques designed for IoT, e.g., lightweight cryptographic algorithms [53].

1.2 Techniques and Applications of IoT

The IoT systems make good use of a collection of cutting-edge technologies and have been widely used in many applications [17]. Typical IoT applications include smart home [7, 16, 27], smart city [43, 57], smart logistics and the related areas [58, 61], health care [15, 22], industry [8], which makes the industrial Industrial of Things (IIoT), environmental monitoring [24], and even smart space [3].

There are many review and survey papers in IoT, some are focused on the IoT literatures [35], on the technical survey [2], some are about IoT development survey [45, 55], some are about the IoT security [1, 11, 38], some are about IoT research directions [50].

Among the IoT research topics, the IoT architecture is a fundamental issue. There have been a number of IoT architectures proposed [14, 26, 37, 46]. A commonly accepted IoT architecture is a 3-layer architecture, where the 3 layers are: the perception layer (which is also known as the sensor layer), the network layer, and the processing layer (which is sometimes called the application layer). This basic IoT architecture as well as other architectures will be further studied in Chap. 2.

So far, many IoT security problems have been extensively studied. Kozlov et al. studied the security and privacy problems in IoT architectures [30], Matharu et al. studied the security issues in IoT [36], Ashraf and Habaebi studied the IoT threat mitigation problem [5], Babar et al. studied the embedded security framework for IoT [6], Shuhaimi et al. studied the SDN approach for IoT security [47].

1.3 The Components of an IoT System

So far there is no precise definition of IoT systems. However, the concept of IoT has been commonly understood as having the following functionalities: data collection, data transmission, data processing and application. The process of data collection can be done by sensors that collect the environmental data, or by RFIDs that can tell the identity information of the labeled goods, or by controllers, actuators and adjustors that can change the condition of a physical device or a physical switch. The process of data transmission means that data are transmitted over wide area networks, and some localized data transmission is treated as within the process of data collection. The process of data processing is known as centralized data processing, which is typically done in a cloud computing platform, including the infrastructure of the platform and the services provided by the platform. Data storage, data processing (e.g. data encryption, data freshness verification, identity authentication), other functionalities (e.g. platform, infrastructure, software) are all treated as services, and a specific application is done by a collection of the services provided by the data processing center (e.g. cloud computing platform), hence the application is often understood as part of data processing.

1.4 IoT Industries and Their Characteristics

Different IoT industries have different characteristics. Commonly known IoT industries include smart logistics, intelligent transport system, smart home, smart medical system (known as *wise information technology of 120, WIT120*), smart power grid, and smart city. The term "smart" is often replaced by "intelligent". A few well-know IoT industries are described in this chapter to show their characteristics.

1.4.1 Characteristics of Mart Logistics

Smart logistics is a system that uses advanced techniques, including RFID systems, wireless sensor networks, mobile communications, to provide intelligent services such as storage of goods, automatic distribution of goods, source tracing of products, vehicle positioning, tracking and tracing of vehicles and of people. Smart logistics aims to save the cost of goods delivery and providing higher quality services.

Since smart logistics is mainly about information processing about goods delivery, it is understandable that RFID systems play a very important role in the smart logistics. RFID systems provide a convenient way to label and identify goods efficiently and conveniently. RFID tags have some computing and communication capabilities, this feature enables information security protection can be done for RFID systems, typically for the communication between an RFID tag and a reader.

Apart from RFID systems, another system widely used in smart logistics is Global Positioning System (GPS). Now an alternative of GPS-like system is the Beidou Navigation Satellite System (BNSS), both of the systems use satellites to allocate the position of the endpoint devices, and are suitable for outdoor devices. In smart logistics, indoor positioning techniques do not have much attraction, so the outdoor positioning systems are sufficient for practical applications.

Security problems in RFID systems include illegal cloning, relay attacks, RFID illegal tracking (which is regarded as RFID privacy), and RFID authentication (must be lightweight). Bunch RFID identification and authentication are new challenges that need to be further studied.

1.4.2 Characteristics of Intelligent Transport Systems

Intelligent transport system is an IoT system with endpoint devices collecting traffic information, location information, speed information, etc., such information is transmitted to a database center for smart processing. Intelligent transport systems are meant to provide a variety of transport services, including traffic information with dynamic updating, smart navigation, intelligent management of vehicles, intelligent message exchange between moving vehicles, and smart parking services. Information of public transport is also a specific branch of intelligent transport system.

A typical example of intelligent transport system is the electronic toll collection (ETC) system. ETC system is a practical application of RFID, where the user endpoint device is an RFID tag, and the toll collection point is a reader/writer. Another example of intelligent transport is smart parking system, where available parking spaces can be seen from an user endpoint device, and drivers can book a parking space before parking, avoiding the embarrassing situation when more people drive their vehicles to a same parking space. In some automatic parking systems, vehicles are moved to an available parking space automatically. Apart from what mentioned above, intelligent transport systems also include the following applications: (1) Dynamically updated public transport information made available to people, who can better manage their schedule and transport choices; (2) Vehicles in an automatic driving system to interact with each other, so that road can be effectively used; (3) Vehicle management and services over the networks, such as dealers serving their customers, or a vehicle rental agent managing and controlling their vehicles.

The RFID tag cloning is a typical security in intelligent transport systems. With a cloned vehicle plate, an unregistered vehicle can impersonate another registered one, and traffic rule breach could be wrongly directed to other innocent people. With the development of intelligent transport systems, IT security and OT security techniques will be widely used.

1.4.3 Characteristics of Smart Home

Smart home is such a system where smart electric and electronic appliances such as audio/video equipment, lighting system, air conditioner control system, security system, are connected to the network, hence the home residents can control those appliances over a user mobile device, and smart cooperative way of working can be made possible.

Fire sensors, smoke sensors, gas sensors, and video surveillance systems, are common parts in a smart home system. Data collected by different sensors and surveillance systems are transmitted to a database center (in the processing layer), where the transmission can be made via wired or wireless networks (specifically mobile communication networks, e.g. 3G, LTE/4G, or 5G). The end users can get access to those data to know the home condition, and can send commands to operate specific appliances (e.g. to turn on a rice cooker, an air conditioner, or a heater). Smart home systems are not restricted to be used in residential areas, they can be used in other public areas such as supermarkets, shopping complexes, office buildings.

Regardless whether a smart home system is used by a residential house, or a public area, environmental data need to be transmitted to a data processing center and stored in a database. During data transmission, it can be eavesdropped or illegally modified. In the smart home environment, eavesdropping of communication signals is easy, because most of the signals are transmitted over wireless networks. Illegal

modification of authorized data is difficult, because the attacher could hardly shield the original signals and send modified ones. However, forgery attack is possible if the data has no security protection.

Data eavesdropping can cause privacy and confidentiality problems, e.g., electricity consumption of a specific household can be revealed, and home monitoring data can be accessed. Illegal modification of data or forgery attack can cause even severer problems, e.g., wrong operation of some switches or electric appliances. Therefore, a smart home system needs proper security services, including privacy protection, data confidentiality protection, data integrity protection, data freshness verification, and identity authentication, including the identities of users and those of IoT devices in the smart home system.

1.4.4 Characteristics of Intelligent Medical System/WIT120

Intelligent medical system is also known as WIT120, and the latter term is more popular. WIT120 is a complex medical service system supported by cutting-edge information technology, targeting at providing more precise, more prompt, and more intelligent medical treatment, and in the mean time, WIT120 is supposed to release doctors and nurses from heavy workload.

A WIT120 system may contain the following sub-systems: Hospital Information System (HIS), Laboratory Information Management System (LIS), Picture Archiving and Communication System (PACS), Electronic Health Record (EHR), Intelligent Hospital System (e.g., make an appointment), Home Caring System.

As a specific application of WIT120, home caring system and medical treatment system use some medical sensors to collect and analyze some health parameters of patients such as blood pressure, blood glucose, and heart rates. Such data are collected by the medical sensors and transmitted to a data processing center for further analysis, and the results are sent to a monitoring end. When some of the health parameters show abnormal condition, some doctors from a local service center are arranged to come and see the patients and provide face to face treatment. In some urgent cases, a command of instruction may be sent to a specific wearable medical device to do some emergency response (e.g. heart pacemaker). Since the health parameters are personal information of the patients, privacy protection is an important security measure in WIT120.

1.5 Security and Privacy Issues in IoT

Security and privacy are preliminary security requirements in network environment. This is the same situation in the IoT era. In fact, security and privacy issues become even severer in IoT systems. In Internet environment, security and privacy issues are only related to digital data. However, in IoT applications, security and privacy

issues are related to the digital data and physical devices as well. Malicious control of some medical devices can cause threat to peoples' health, life, and malicious intrusion into self-driving vehicles may cause social safety problems.

In IoT applications, IT security has some characteristics, including the following:

Lightweight. Cryptographic algorithms for IoT systems should be lightweight, because many IoT devices are resource constrained.

Although there are also many IoT devices with sufficiently large computation and communication capabilities, those devices can be treated as traditional information systems, in such case, traditional IT security techniques can be properly used.

However, there are also a large number of resource-constrained IoT devices. Traditional security techniques do not pay particular attention to resource-constrained environments, particularly some extremely resource-constrained environments (e.g., RFID tags). Therefore, IoT security techniques should pay attention to the resource-constraint feature of implementation environments. devices

Data freshness. Data freshness is an important security measure in IoT applications, it is closely related to replay attack.

In the Internet environment, data freshness is not an issue, because data is often split into parts for transmission. Some parts of the data may arrive in replicates, and this is not an issue in Internet environment. However, in IoT applications, if the data arrives in replicates, then each copy of the data may be processed, this makes replay attack possible. For example, if a command is replayed in a wrong time, then the execution of the command may cause a big damage. The security service of data freshness is to detect replay attack, and is specifically important in IoT applications.

There can be other features of IT security in IoT applications, and the above are representative ones.

Network security requirement is not restricted to data content protection. In fact, practical IoT systems pay more attention to physical devices. This is particularly important in industrial IoT (IIoT), where OT security is more important than IT security. The OT security has the following features:

Validity. The data form should be correct. This is related to command data, which is usually of a specific form, e.g., format of data or the range of a numerical value. It should be noted that the data validity is different from data integrity, The former is about the logical format of data, and the latter is about illegal modification of the data. Validity is a specific feature of OT security, and the data integrity is an IT security service.

Authenticity. It is to make sure that the data comes from an authorized source. Authenticity means that the data itself is valid, i.e., the data is not replayed, not spoofed, not illegally modified, and comes from an authentic source.

Being genuine. Even though the data is authentic, it may be manipulated by an attacker who controls an authorized system. Traditional IT security techniques can hardly be effective in judging whether the data is genuine, particularly when

the data is verified as being authentic. This is a specific characteristic of OT security, which has some essential differences from IT security. This problem will be further discussed in Chap. 11.

Apart from the above described IT security characteristics and OT security features of IoT systems, other specific security characteristics and features of IoT systems are as follows:

Privacy issues. Privacy is known as a personal information security issue. Many IoT applications need personal information, for example, smart home system and smart medical system. In such systems, personal identity information should be kept secure to the public. This is identity privacy problem. Another privacy problem is location privacy, which is about keeping the location information of someone or something unknown to the public, it is another specific security issue in many IoT applications such as smart transport and smart logistics.

Not to become an attacker. Traditional information system security only considers how to protect the target object (e.g., a systems, data, communication protocol) against external attack. With respect to IoT devices, it is possible that large amount of IoT devices form a botnet to lunch DDoS attacks to other devices and service provision platforms on the Internet. Although the DDoS attacks are made by attachers who can control the IoT devices, e.g., by intrude into them, the IoT device manufactures are responsible for the security vulnerability of their devices. In this sense, how to ensure IoT devices not to become attackers is a specific security measure in IoT applications.

The above characteristics of IoT security indicate that many traditional network security techniques are not suitable for the IoT systems, they need to be modified, or new security techniques targeting at IoT systems are designed. Fortunately, some lightweight cryptographic algorithms [10, 56, 59] have been designed targeting at IoT applications, international conferences specializing in lightweight cryptography have been organized, NIST investigated the need for lightweight cryptographic algorithms, and has published a call for algorithms (test vector generation code) to be considered for lightweight cryptographic standards [12]. This indicates that lightweight cryptographic algorithms are likely to become new members of international standards.

While lightweight cryptographic algorithms provide useful tools for IoT security services, in real network-based applications, security services are often provided by security protocols. Hence, lightweight security protocols are of more practical significance. It should be pointed that, in IoT applications, data confidentiality is not necessarily provided by using an encryption algorithm, data integrity is not necessary to use a message authentication code algorithm, and identity authentication does not necessarily need a challenge-response protocol or a digital signature. Such features can be reflected from the example in Sect. 12.6.

1.6 Summary

The wide IoT applications cover a large variety of different industries. IoT application systems have no geographical restrictions, they can be geographically across cities or even countries, may involve many cutting-edge technologies. Among the technologies, security techniques play an essential role enabling IoT applications to be safe and beneficial.

This chapter introduces some basic IoT related concepts and some specific IoT applications. Security concerns and characteristics of security requirements for IoT systems are discussed.

It was pointed out that IoT security has some special characteristics that have essential differences from traditional information security. IoT security includes two aspects, IT security and OT security. With respect to IT security, lightweight is the most obvious characteristic; With respect to OT security, how to protect an IoT system when it might be controlled by an intruder is a special issue; With respect to privacy, identity privacy and location privacy are important security issues in different IoT applications; Another security requirement of IoT systems is that, IoT devices should not be involved in attacking other resources. Further study on the IoT security problems and techniques will be introduced in other chapters.

With respect to the security and privacy issues, further study on privacy problems in IoT applications can be found in e.g. [4, 28, 39, 40, 44, 54], Identity management and authentication problems can be found in [20, 21, 29, 42]. Other IoT security problems are also extensively studied in [32–34, 41, 48, 51]. For the status about IoT security research, readers are referred to overview articles (see e.g., [18, 19, 25, 31, 52, 60]).

References

1. F.A. Alabaa, M. Othmana, I.A.T. Hashema, F. Alotaibi, Internet of things security: a survey. J. Netw. Comput. Appl. **88**, 10–28 (2017)
2. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surveys Tutor. **17**(4), 2347–2376 (2015)
3. S. Amendola, R. Lodato, S. Manzari, C. Occhiuzzi, G. Marrocco, RFID technology for IoT-based personal healthcare in smart spaces. IEEE Int. Things J. **1**(2), 144–152 (2014)
4. O. Arias, K. Ly, Y. Jin, Security and privacy in IoT era, in ed. by H. Yasuura, et al., *Smart Sensors at the IoT Frontier* (Springer, Berlin, 2017), pp. 351–378
5. Q. M. Ashraf, M.H. Habaebi, Autonomic schemes for threat mitigation in internet of things. J. Netw. Comput. Appl. **49**, 112–127 (2015)
6. S. Babar, A. Stango, N. Prasad, J. Sen, R. Prasad, Proposed embedded security framework for Internet of Things (IoT). in *Proceedings of the 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, Chennai (2011), pp. 1–5
7. A. Bako, A. Ali, Cyber and physical security vulnerability assessment for IoT-based smart homes. Sensors **18**(3), 817 (2018). https://doi.org/10.3390/s18030817

8. Z. Bi, Y. Liu, J. Krider, J. Buckland, A. Whiteman, D. Beachy, J. Smith, Real-time force monitoring of smart grippers for Internet of Things (IoT) applications. J. Ind. Inf. Integr. **11**, 19–28 (2018)
9. K. Boeckl, M. Fagan, W. Fisher, N. Lefkovitz, K.N. Megas, E. Nadeau, D.G. O'Rourke, B. Piccarreta, K. Scarfone, Considerations for managing Internet of Things (IoT) cybersecurity and privacy risks, NISTIR 8228. https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8228.pdf (2020-02-07 successfully visited)
10. A. Bogdanov, L.R. Knudsen, G. Leander, et al., PRESENT: An ultra-lightweight block cipher, in *Proceedings of 9th International Workshop on Cryptographic Hardware and Embedded Systems - CHESS 2007*. Lecture Notes in Computer Scienc, vol. 4727 (Springer, Berlin, 2007), pp. 450–466
11. S. Cirani, G. Ferrari, L. Veltri, Enforcing security mechanisms in the IP-based Internet of things: an algorithmic overview. Algorithms **6**(2), 197–226 (2013)
12. CSRC's project on lightweight cryptography, https://csrc.nist.gov/projects/lightweight-cryptography. Visited on 12 Jan 2020
13. M. Darianian, M.P. Michael, Smart home mobile RFID-based Internet-of-things systems and services, in *Proceedings of 2008 International Conference on Advanced Computer Theory and Engineering*, Phuket (2008), pp. 116–120. https://doi.org/10.1109/ICACTE.2008.180
14. D.G. Darwish, Improved layered architecture for internet of things. Int. J. Comput. Acad. Res. **4**(4), 214–223 (2015)
15. Y.J. Fan, Y.H. Yin, L.D. Xu, Y. Zeng, F. Wu, IoT-based smart rehabilitation system. IEEE Trans. Ind. Inf. **10**(2), 1568–1577 (2014)
16. C. Gao, Z. Ling, Y. Yuan, The research and implement of smart home system based on Internet of things, in *Proceedings of the 2011 International Conference on Electronics, Communications and Control (ICECC)*, Ningbo (2011), pp. 2944–2947
17. M. Gigli, S. Koo, Internet of things: services and applications categorization. Adv. Int. Things **1**(2), 27–31 (2011)
18. J. Granjal, E. Monteiro, J.S. Silva, Security for the internet of things: a Survey of existing protocols and open research issues. IEEE Commun. Surveys Tutor. **17**(3), 1294–1312 (2015)
19. K. Gupta, S. Shukla, Internet of things: Security challenges for next generation networks, in *Proceedings of the 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, Noida (2016), pp. 315–318
20. S. Horrow, A. Sardana, Identity management framework for cloud based Internet of things, in *Proceedings of the First International Conference on Security of Internet of Things*, Kollam (2012), pp. 200–203
21. C. Hu, J. Zhang, Q. Wen, An identity-based personal location system with protected privacy in IoT, in *Proceedings of the 4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, Shenzhen (2011), pp. 192–195
22. S.R. Islam, D. Kwak, M.H. Kabir, M. Hossain, K.S. Kwak, The internet of things for health care: a comprehensive survey. IEEE Access **3**, 678–708 (2015)
23. ITU Internet Reports 2005, The Internet of Things, http://www.itu.int/pub/S-POL-IR.IT-2005/e
24. A.R. Jaladi, K. Khithani, P. Pawar, K. Malvi, G. Sahoo, Environmental monitoring using wireless sensor networks (WSN) based on IoT. Int. Res. J. Eng. Technol. **4**(1), 1371–7378 (2017)
25. Q. Jing, A.V. Vasilakos, J. Wan, J. Lu, D. Qiu, Security of the Internet of things: Perspectives and challenges. Wirel. Netw. **20**, 2481–2501 (2014)
26. R. Khan, S.U. Khan, R. Zaheer, S. Khan, Future internet: The internet of things architecture, possible applications and key challenges, in *Proceedings of the 2012 10th International Conference on Frontiers of Information Technology (FIT)*, Islamabad (2012), pp. 257–260
27. I.U. Khan, M.U. Shahzad, M.A. Hassan, Internet of things (IoTs): applications in home automation. Int. J. Sci. Eng. Adv. Technol. **5**(1), 79–84 (2017)

28. B. Khoo, RFID as an enabler of the internet of things: Issues of security and privacy, in *Proceedings of the 2011 International Conference on Internet of Things (iThings/CPSCom) and 4th International Conference on Cyber, Physical and Social Computing*, Dalian (2011), pp. 709–712
29. N. Koshizuka, K. Sakamura, Ubiquitous ID: Standards for ubiquitous computing and the internet of things. IEEE Pervasive Comput. **9**, 98–101 (2010)
30. D. Kozlov, J. Veijalainen, Y. Ali, Security and privacy threats in IoT architectures, in *Proceedings of the 7th International Conference on Body Area Networks (BodyNets'12)*, Oslo (2012), pp. 256–262
31. S.A. Kumar, T. Vealey, H. Srivastava, Security in internet of things: Challenges, solutions and future directions, in *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, Koloa (2016), pp. 5772–5781
32. Z. Li, X. Yin, Z. Geng, H. Zhang, P. Li, Y. Sun, H. Zhang, L. Li, Research on PKI-like protocol for the internet of things, in *Proceedings of the 5th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, Hong Kong (2013), pp. 915–918
33. J. Liu, Y. Xiao, C.P. Chen, Authentication and access control in the internet of things, in *Proceedings of the 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Macau (2012), pp. 588–592
34. W. Liu, L. Zhang, Z. Zhang, C. Gu, C. Wang, XOR-based low-cost reconfigurable PUFs for IoT security. ACM Trans. Embed. Comput. Syst. **18**(3), 1–21 (2019)
35. S. Madakam, R. Ramaswamy, S. Tripathi, Internet of things (IoT): a literature review. J. Comput. Commun. **3**, 164–173 (2015)
36. G.S. Matharu, P. Upadhyay, L. Chaudhary, The internet of things: Challenges & security issues, in *Proceedings of the 2014 International Conference on Emerging Technologies (ICET)*, Islamabad (2014), pp. 54–59
37. Y. Miao, Y.X. Bu, Research on the architecture and key technology of internet of things (IoT) applied on smart grid, in *Proceedings of the 2010 International Conference on Advances in Energy Engineering (ICAEE)*, Beijing (2010), pp. 69–72
38. N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, N. Ghani, Demystifying IoT security: an exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations, IEEE Commun. Surveys Tutor. **21**(3), 2702–2733 (2019)
39. A.A. Nezhad, A. Miri, D. Makrakis. Location privacy and anonymity preserving routing for wireless sensor networks. Comput. Netw. **52**(18), 3433–3452 (2008)
40. F. Nizzi, T. Pecorella, F. Esposito, L. Pierucci, R. Fantacci, IoT security via address shuffling: the easy way. IEEE Int. Things J. **6**(2), 3764–3774 (2019)
41. J.R. Nurse, A. Erola, I. Agrafiotis, M. Goldsmith, S. Creese, Smart insiders: Exploring the threat from insiders using the Internet-of-things, in *Proceedings of the 2015 International Workshop on Secure Internet of Things (SIoT)*, Vienna (2015), pp. 5–14
42. N. Park, N. Kang, Mutual authentication scheme in secure internet of things technology for comfortable lifestyle. Sensors **6**(1), 20–20 (2016)
43. C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Sensing as a service model for smart cities supported by Internet of things. Transa. Emerg. Telecommun. Technol. **25**(1), 81–93 (2014)
44. R. Roman, J. Zhou, J. Lopez, On the features and challenges of security and privacy in distributed Internet of Things. Comput. Netw. **57**(10), 2266–2279 (2013)
45. O. Said, M. Masud, Towards internet of things: Survey and future vision. Int. J. Comput. Netw. **5**(1), 1–17 (2013)
46. P. Sethi, S.R. Sarangi, Internet of things: architectures, protocols, and applications. J. Electr. Comput. Eng. **2017**, Article ID 9324035, 25 (2017)
47. F.A. Shuhaimi, M. Jose, A.V. Singh, Software defined network as solution to overcome security challenges in IoT, in *Proceedings of the 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida (2016), pp. 491–496

48. S. Sicari, A. Rizzardi, L.A. Grieco, A. Coen-Porisini, Security, privacy and trust in Internet of Things: the road ahead. Comput. Netw. **76**, 146–164 (2015)
49. A.K. Sikder, G. Petracca, H. Aksu, T. Jaeger, A.S. Uluagac, A survey on sensor-based threats to Internet-of-things (IoT) devices and applications. Cryptogr. Security (cs.CR) (2018). arXiv:1802.02041
50. J.A. Stankovic, Research directions for the internet of things. IEEE Int. Things J. **1**(1), 3–9 (2014)
51. B.V. Sundaram, M. Ramnath, M. Prasanth, V. Sundaram, Encryption and hash based security in Internet of things, in *Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai (2015), pp. 1–6
52. H. Suo, J. Wan, C. Zou, J. Liu, Security in the Internet of things: A review, in *Proceedings of the 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*, Hangzhou, vol. 3 (2012), pp. 648–651
53. T. Suzaki, K. Minematsu, S. Morioka, E. Kobayashi, TWINE: a lightweight block cipher for multiple platforms, in *Selected Areas in Cryptography - SAC2012*. Lecture Notes in Computer Science, vol. 7707 (Springer, Berlin, 2013), pp. 339–354
54. H. Tao, W. Peiran, Preference-based privacy protection mechanism for the Internet of things. in *Proceedings of the 2010 International Symposium on Information Science and Engineering (ISISE)*, Shanghai (2010), pp. 531–534
55. A. Whitmore, A. Agarwal, L.D. Xu, The internet of things - a survey of topics and trends. Inf. Syst. Front. **17**, 261–274 (2015)
56. W. Wu, L. Zhang, LBlock: a lightweight block cipher, in *Proceedings of 9th International Conference on Applied Cryptography and Network Security*. Lecture Notes in Computer Science, vol. 6715 (Springer, Berlin, 2011), pp. 327–344
57. A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities. IEEE Int. Things J. **1**(1), 22–32 (2014)
58. M. Zhang, T. Yu, G.F. Zhai, Smart transport system based on the internet of things. Appl. Mechan. Mater. **48**, 1073–1076 (2011)
59. W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, I. Verbauwhede, RECTANGLE: a bit-slice ultra-lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. **58**(12), 1–15 (2015)
60. Z.K. Zhang, M.C.Y. Cho, C.W. Wang, C.W. Hsu, C.K. Chen, S. Shieh, IoT security: Ongoing challenges and research opportunities. in *Proceedings of the IEEE 7th International Conference on Service-Oriented Computing and Applications (SOCA)*, Matsue (2014), pp. 230–234
61. Z. Zhou, Z. Zhou, Application of internet of things in agriculture products supply chain management, in *Proceedings of the 2012 International Conference on Control Engineering and Communication Technology (ICCECT)*, Liaoning (2012), pp. 259–261

Chapter 2
Architectures of the Internet of Things

This chapter introduces some common architectures of Internet of Things (IoT). A number of IoT architectures with layered structures are introduced, and a 6-domain IoT architecture proposed in a China standard is described, which is designed based on different types of services. This chapter discusses the relations between different IoT architectures. The discussions of the relations how that, different IoT architectures do not have substantial differences, they are different perspectives in viewing an IoT system.

2.1 Introduction

Before the introduction of IoT security techniques, it is important to know what IoT is about. Chapter 1 has presented some introduction about the concept of IoT roughly. This chapter is designed to give a more detailed description about the features of IoT by presenting different architectures and their relations.

Although there is no precise definition about what Internet of Things (IoT) is, the concept of IoT can be properly described by an IoT architecture. As described in Chap. 1, the processes in an IoT system includes data collection, data transmission, and data processing and application. It is undoubtedly that an IoT architecture will include those processes, which is what the most popular three-layer IoT architecture tells.

Apart from the three-layer IoT architecture, there are other architectures of IoT, which essentially describe the same IoT system but from different angles. This chapter describes different architectures of IoT proposed in public literatures.

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_2

2.2 The Basic Three-Layer Architecture of IoT

The concept of IoT has been proposed for decades. However there is no scientific definition of the IoT. It does not seem to have a proper definition, because the content of IoT changes with time. For such a concept, it can be described by its features, which can be treated as a descriptive definition.

The description of IoT can be presented based on an architecture, i.e., what an IoT is composed of. However, from different viewpoint, a number of different IoT architectures have been proposed.

An IoT can be viewed as being composed of different layers of data processing, such a viewpoint forms an architecture of IoT. The most common architecture of IoT is a three-layer architecture, which is composed of a layer of endpoint devices and functionalities, a layer of centralized processing platform and its services, and a layer connected with the former two layers. There are different names of the three-layer architecture, with some minor differences.

2.2.1 IoT Architecture Based on Data Flow: A Basic IoT Architecture

Based on how the IoT processes data, the IoT can be treated as having three logical layers: perception layer, network layer, and processing layer (which is sometimes called the application layer), as shown in Fig. 2.1.

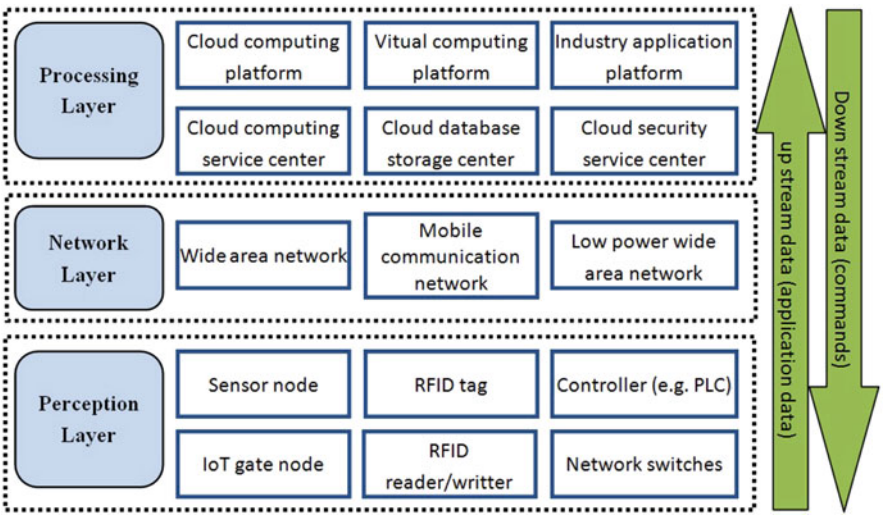

Fig. 2.1 The basic three-layer architecture of IoT

From Fig. 2.1 it is seen that, the perception layer is composed of sensors (including video surveillance), IoT gate nodes, RFID tags, RFID readers/writers, network switches and controllers. The perception layer also includes other kinds of devices and local networks. The main task of the perception layer is to collect environmental information, working status information and identity information of IoT devices, and to execute commands from the processing layer. Human direct interaction with the perception layer devices is also possible, however this manner of operation is not a specific feature of IoT.

2.2.1.1 The Perception Layer

As is shown in Fig. 2.1, the perception layer of IoT is not just about RFID, it includes the following kinds of devices:

1. Endpoint sensing devices, including sensors of all kinds, network video cameras, GPS devices, which are used to collect environmental data;
2. Tagged things, mainly about physical things that do not have communication capabilities. Those things are identified by an unique identifier attached to them, in the form of radio-frequency identification (RFID), bar code, or QR code. These identifiers are not independent devices by themselves, but used to identify the things that they are attached to;
3. Controllable and operational devices, including PLCs, networked controllers and actuators, networked vehicles, networked unmanned aerial vehicles (UAVs), networked switches, RFID readers and writers, Electronic toll collection (ETC) readers. These devices receive commands sent over the network and execute the commands, and may send responses back;
4. Information display and interaction devices, including transport information display screens, smart grid charging hubs, community smart charging piles;
5. Communication devices, including IoT gate devices (known as Sink nodes as in sensor networks), network routers and signal amplifiers;
6. User endpoint devices. Such devices are not only to display information, but get feedback from users to the data processing center, or even back to the endpoint IoT devices. A typical example is when a user operates on a mobile phone (as a user endpoint device) to instruct the action of a home-based heater (as an endpoint IoT device).

Apart from the physical devices, the perception layer of IoT also include some local area networks, including wired LANs (e.g., Ethernet, Fieldbus, Profibus) and

wireless LANs (e.g., ZigBee,[1] bluetooth, WiFi, LORA.[2]) These networks are used to connect the IoT devices in the perception layer locally.

With the development of IoT, there will be more devices classified as being in the perception layer. Sometimes a same kind of device can be classified as a perception layer device, or a network layer device, or an application layer device, depending on the functionalities and services that it provides.

For convenience of description, the IoT devices that transmit data to the data processing center are called *IoT gate devices* or *IoT gate nodes* (or IoT gates for short), and call the IoT devices that collect the environmental data or the identity information, or execute down-stream commands, as *endpoint IoT devices* (e.g., sensor nodes). It can be seen that the transmission of data from a sensor node to the data processing center needs the help of an IoT gate node. However, some IoT devices capture environmental information and transmit directly to the data processing center, such as Webcam, such a device is actually an integration of a sensor node and an IoT gate node, a device with two functionalities and one physical body.

It should be noted that, the above classification of IoT devices is just for references and for easy understanding, there may be different classifications. For example, RFID readers may be classified as a endpoint device instead of an IoT gate. Nevertheless, different classification has little effect on the design of IoT application systems, and essentially no effect on the design of security solutions.

2.2.1.2 The Network Layer

The network layer of IoT describes means for long distance transmission of data, from the data collection venue to a data processing center. So the network layer is composed of traditional wide area networks (WANs) and wireless WANs. Recently a new type of wireless WAN is developed targeting at serving IoT applications, they are low power wide area networks (LPWANs). Typical LPWAN networks include NB-IOT, LoRa, SigFox, they have been widely constructed with commercialized operation.

The network layer is composed of more than one network architectures. For example, data may be transmitted first using the mobile communication network, then the wired Internet.

[1] ZigBee is also treated as personal area network (PAN), here the LAN is in the sense of a broad meaning, including all the short distance networks that can be used in IoT.

[2] Although LORA was initially designed as a wide area network with low power, in order to cooperate with the NB-IOT network, LORA can be configured as a wireless LAN.

2.2.1.3 The Processing Layer

The processing layer is sometimes also called application layer, because this layer provides services and data for different applications. The processing layer is typically implemented by a cloud computing platform, including platform infrastructure (e.g., hardware, software, operation system, Hadoop), computing services (e.g., virtualization service, NaaS-network as a service, SaaS-Software as a service, SecaaS-security as a service), and database services (e.g., MySql, NoSQL).

The processing layer is also responsible for managing end users to access the relevant data, to get required services, including data service and computing service, and is linked to various kinds of applications. Hence the processing layer can further be split into processing layer (responsible for data processing) and application layer (responsible for user management, service provision, and cooperation with various applications).

In the processing layer of IoT, the techniques of smart processing and intelligent computing can be found throughout the processing layer, from how data is stored and managed, how the system is virtualized, how users are managed, and how rich services are provided.

When data is transmitted from the perception layer to the processing layer, it is mostly the application data, and the data transmission is called up-stream data transmission. When data is transmitted from the processing layer to the perception layer, it is mostly a kind of command data, which is used to instructing endpoint devices in the perception layer how to operate. This way of data transmission is called down-stream data transmission. Both the down-stream data transmission and the up-stream data transmission will need to go through the network layer.

The processing layer of IoT is mostly realized via cloud computing, including cloud computing platforms and services that they provide. There are traditional computing platforms as small scale data processing platforms can be used as data processing of IoT systems. Since traditional computing platforms have some disadvantages compared with cloud computing, including being less flexible in scalability, less reliability, and perhaps low level of security, the study on the processing layer is mainly concerned with cloud computing platforms and their services.

2.2.2 Where to Place Fog Computing and Edge Computing

While the concept of IoT is new, the content and involved technologies are not brand new. Most of the technologies in IoT are evolutionally developed from existing ones, and IoT systems have a more sophisticated integration and higher level of requirement (e.g., higher efficiency, low energy consumption, more restriction on the execution environment) of the existing techniques and certain new techniques. With respect to the perception layer, sensors capture the environmental information independently or collaboratively over a LAN, and conduct primitive processing

before the sensor data is sent to the processing layer. Hence, the information from the perception layer may not be the raw data about the environment, position, or device condition. More advanced computing services for the perception layer and beyond are called *fog computing* and *edge computing*. *Fog computing* is a decentralized computing infrastructure in which data, compute, storage and applications are located somewhere between the data source and the cloud, and *edge computing* is a distributed information technology (IT) architecture in which client data is processed at the periphery of the network, as close to the originating source as possible. Many people use the terms of fog computing and edge computing interchangeably, it seems that the edge computing a bit more closer to the perception layer (somehow within the perception layer) than the fog computing.

2.3 IoT Architecture Based on IoT Devices: The Sea-Cloud Architecture

There is a different way to establish an IoT architecture, from the viewpoint of different types of IoT devices. This IoT architecture is called "sea-cloud" architecture [7], where the term "sea" means the tremendous amount of IoT endpoint devices, including sensor devices and users' application devices, the term "cloud" means the cloud computing, including cloud computing platform, cloud storage, and cloud computing services. Between the "sea" and the "cloud", data transmission is provided by networks, this yields a "sea-network-cloud" architecture, or "sea-cloud" architecture for short.

Alternatively, the "sea-network-cloud" architecture is also called "cloud-tunnel-endpoint" architecture, where the "cloud" has the same meaning in both of the architectures, the "tunnel" means how data is transmitted, corresponding to the term "network", and the "endpoint" means the IoT endpoint devices, including sensor devices and users' application devices, corresponding to the term "sea". Since it is simply a different name of the same architecture, we will ignore this term of architecture, and treat this architecture the same as "sea-cloud" architecture, as shown in Fig. 2.2.

In an IoT system, the processing layer may encounter enormous large scale of data, called *big data*. The concept of big data means that the data is too large or too complicated to be properly processed by traditional data processing techniques. Hence, the processing layer of IoT usually means the cloud computing, and is named as "cloud" for short.

The relation between the architecture shown in Fig. 2.2 and the one shown in Fig. 2.1 is as follows: The "network" layer is the same in both of the architectures. The "cloud" layer in the architecture shown in Fig. 2.2 is almost the same as the "processing" layer in the architecture shown in Fig. 2.1, except that they are viewed from different angles. The "sea" layer of the architecture as shown in Fig. 2.2 is the same as the "processing layer" of the three-layer architecture.

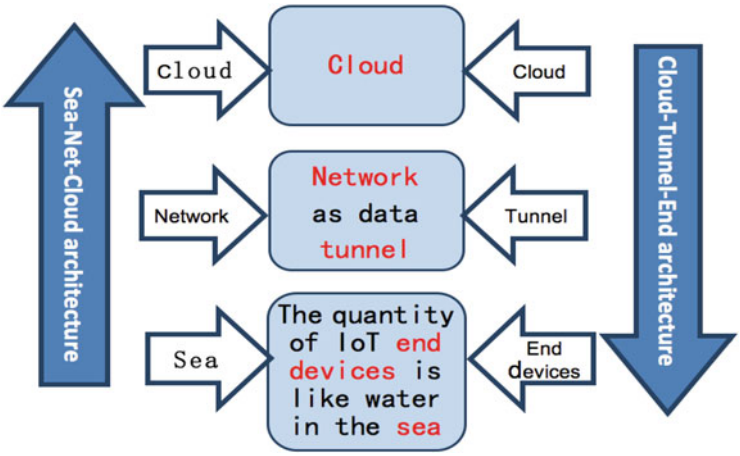

Fig. 2.2 The sea-cloud architecture of IoT

It is seen that the two IoT architectures shown in Figs. 2.1 and 2.2 are essentially about the same thing. The basic three-layer IoT architecture is based on the order about how the data is processed, while the sea-cloud IoT architecture is based on the classification of physical devices in an IoT system. Both of the above mentioned architectures are based on the visible components. Some of the invisible things such as computing services are important components in IoT, but not considered in the above IoT architectures. The visible components in IoT are mostly static, and the invisible components are usually dynamic.

2.4 A Four-Layer Architectures of IoT

As mentioned above, the basic three-layer architecture of IoT can be changed into a four-layer architecture by splitting the processing layer into two layers. Without confusion, the two new layers are called processing layer (same name, but a sub-layer of the processing layer in the basic three-layer architecture) and application layer. The four-layer architecture can be shown in Fig. 2.3.

The processing layer in the four-layer architecture is mainly about the cloud platform and the functionalities in the cloud platform, and the application layer is related to actual IoT applications, including how the processed data is used, how users can access the application data and how to request services from the cloud platform (the processing layer), how users are managed, and how users are authenticated.

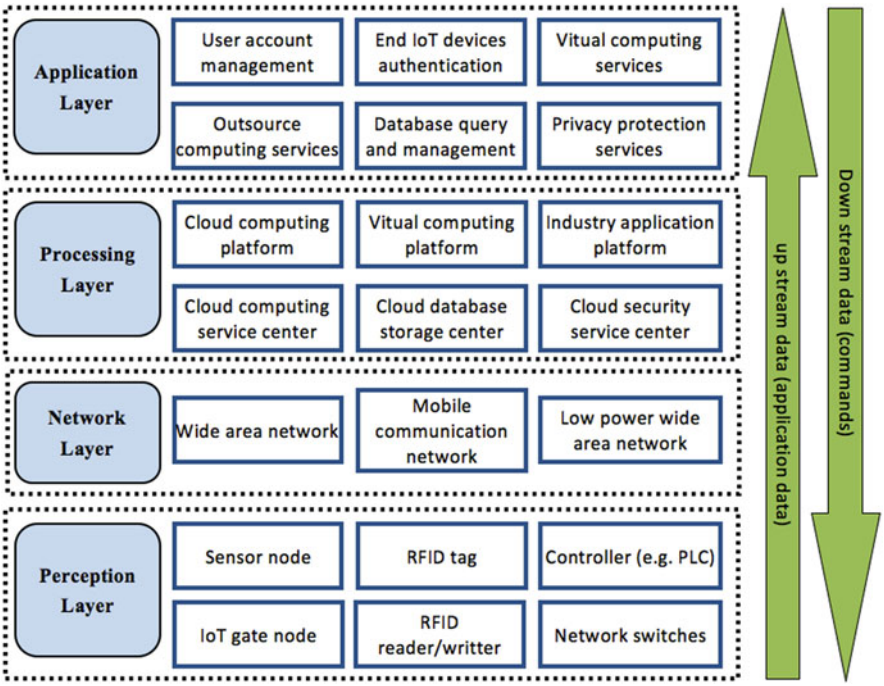

Fig. 2.3 A four-layer architecture of IoT

2.5 A Five-Layer Architecture of IoT

Further splitting the IoT architecture into more layers is possible. In order to further split the functionalities of the application layer, particularly the security functionalities and services, a five-layer architecture was proposed [6]. The five-layer IoT architecture is basically to further split the application layer of the four-layer architecture into application layer and business layer, where the business related applications are grouped into the business layer. The five-layer architecture can be shown in Fig. 2.4.

Although there are other different descriptions of five-layer architecture of IoT (e.g., [3–5]), they are just different ways of splitting the IoT layers.

It is seen that the five-layer architecture has the same coverage as the four-layer architecture about the devices and services, as what are covered by the basic three-layer architecture. Therefore, the basic three-layer architecture plays an important role and is treated as an important reference architecture in this book.

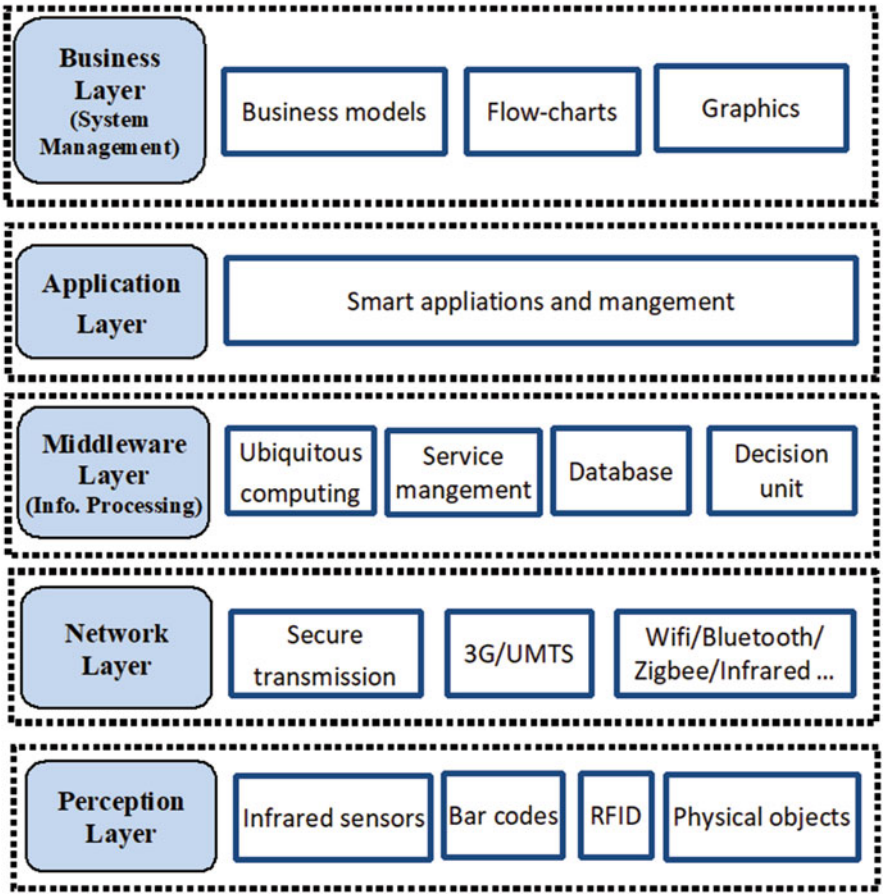

Fig. 2.4 A five-layer architecture of IoT

2.6 A Six-Layer Architecture of Fog Computing

Sethiand Sarangi proposed a six-layer architecture of fog computing [5]. Although it is not an IoT architecture, it forms an important part of IoT and is worth mentioning here. The six-layer architecture inserts monitoring, preprocessing, storage, and security layers between the physical and transport/network layers. The six layers are as follows: Physical layer, Monitoring layer, Preprocessing layer, Storage layer, Security layer, Transport layer. Obviously it is not a complete IoT architecture, it is essentially the architecture of the perception layer, in the sense when the perception layer includes fog computing and edge computing, i.e., the architecture covers all the data collection and data processing before the data is sent to the cloud.

2.7 The Six-Domain Architecture of IoT

In theory, it is possible to further split the five-layer architecture of IoT to form an architecture with more layers. However, such splitting does not seem to be necessary. The basic three-layer architecture and the four-layer architecture seem to be sufficient.

However, there does exist a six-domain architecture of IoT. It is not like the other layered architectures with bottom-up layers [1], but with six domains with inter-correlations. Note that the previous IoT architectures mainly consider the physical devices (a combination of hardware and software) and some static services, it would be helpful to consider an IoT system as being alive, providing services and applications. For this consideration, a six-domain architecture is proposed. Such an architecture was defined in China national standard "GB/T 33474-2016 IoT: an reference architecture", and the six-domain IoT architecture is shown in Fig. 2.5.

For the convenience of description, we call the architecture shown in Fig. 2.5 as *six-domain architecture*. In the six-domain architecture, 6 domains of functionalities of IoT are defined as follows:

User domain. The user domain is a collection of different kinds of users of IoT systems. The normal operation of IoT is for the users as subjects to obtain sensor data from the subject domain, and to send control commands to the subject domain. Users in the user domain include normal users (e.g., customers), industry users (e.g., IoT system operators), government users (e.g., IoT industry supervisors), and other users (e.g., administrators).

Service provision domain. The service provision domain is a collection of all the services provided by IoT applications. It is where the users get services from, including data and operation services, which are defined in specific IoT applications. The service provision domain has no specific physical objects, and has expandability and plurality. With the development of IoT applications in different areas, the actual content of the service provision domain can be dynamically changed.

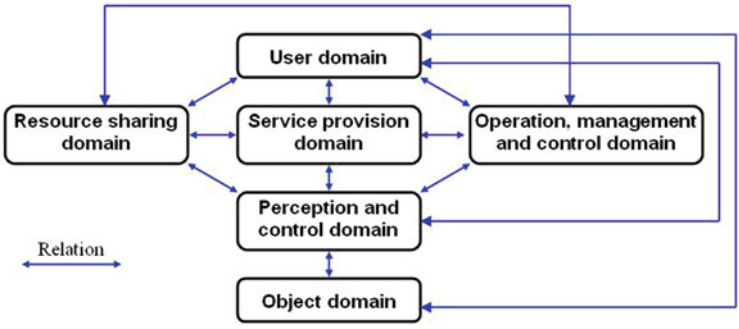

Fig. 2.5 A six-domain architecture of IoT

Perception and control domain. The perception and control domain is responsible for data collection and device operation, including devices and functionalities that can obtain the sensor data or execute control commands. The perception and control domain is the connection between the user domain and the object domain, it is the interface to connect the cyber space and the real physical world. Based on the current techniques, the perception and control domain is closely related to the following sub-systems:

- Wireless sensor networks, which are bases to get sensor data and to execute operation control commands.
- Identification systems, including RFID systems, bar codes and QR codes, which are used to identity physical things that do not have communication capability. A preliminary study on the identity unification problem can be found in [2].
- Positioning systems, including GPS, Beidou Navigation Satellite System, mobile communication positioning system, and other positioning systems. Many mobile IoT devices need to have their position reported to the data processing center in the processing layer.
- Multimedia systems, which are used to capture audio and video data (including voice, picture, digital video) from targeted sources.
- Communication interface systems, which are used for the provision of the communication interface for the target communication objects, and are responsible for the network connection, data processing and data fusion, and protocol conversion. The communication systems may be used to serve for example as power switches, air conditioners, large instruments, and smart digital equipments. In practical applications, the communication interface of smart devices may be integrated into the target communication objects.
- Other systems, that are not yet precisely known at present, and will appear as new components in the perception and control domain.

In a specific IoT application system, according to the specific application requirements, one or more of the above sub-systems may exist. The process of perception and control may interact with each other. The perception and control domain can exist as an independent system to fulfill the task of local perception and operation services.

Object domain. The object domain is a collection of objects from which users want to get some expected data, and those on which users want to operate some control commands. The objects in the object domain can be linked with the sub-systems in the perception and control domain via certain interaction interfaces, which can be for data communication (e.g., COM port, LTP port, USB, and the Ethernet) and for non-data interaction (e.g., physical, chemical, biological interaction, or tag attachment, or position connection).

Resource sharing domain. The resource sharing domain is a collection of platforms and their services on resource exchange and sharing, serving local and external applications. The resource sharing domain includes local resources (e.g., data from perception and control domain and that from service provision domain)

and external resources (e.g., government, industry, and personal information, and data from electronic commerce activities and from commercial transactions).

Operation, management and control domain. The operation, management and control domain (OMC-domain in brief) is a collection of entities that provide IoT system operation, maintenance, management, and legal supervision. The OMC-domain is responsible for technical operation and legal compliance management of IoT systems, ensuring their stable, reliable, and safe operation.

It is easy to see that, the basic three-layer architecture (as well as the four-layer architecture, five-layer architecture) is mostly about the physical composition of IoT, which is a static IoT architecture, while the six-domain architecture is based on the functionalities of an alive IoT system. Although these two kinds of IoT architectures have essential differences, there are some close connections between these two architectures. Some relations of the two architectures are briefly described below, which reflect the correspondence from the functional domains in the six-domain architecture to the layers in the basic three-layer architecture.

1. **User domain.** In the basic three-layer architecture, users are not mentioned. However, it is easy to understand that users will interact with the cloud computing (in the processing layer) and data services. When the basic three-layer architecture turns into actively working status, users will have interaction with the cloud computing, and be managed by the cloud computing platform. In this sense, users in the six-domain architecture are subjects (when sending service requests and operation commands) and objects (when receiving services) of the processing layer of the basic three-layer architecture.
2. **Service provision domain.** By the above description, the service provision domain in the six-domain architecture covers all kinds of services, including some functionalities and services in all the layers of the basic three-layer architecture. The service provision domain is a collection of services provided by all the layers of the three-layer architecture, which can be alternatively viewed as the services provided by all the layers of the four-layer architectures as well as that of the five-layer architecture.
3. **Perception and control domain.** The perception and control domain of the six-domain architecture is about the control functionality of the devices in the object domain, and can be classified as the functionalities in the perception layer of the basic three-layer architecture.
4. **Object domain.** It is easy to see that the objects in the object domain of the six-domain architecture are the devices (e.g., sensors, RFID tags, actuators) in the perception layer of the three-layer architecture. So the object domain of the six-domain architecture is roughly the same as the perception layer of the basic three-layer architecture.
5. **Resource sharing domain.** The resource sharing domain provides services of resource sharing, including data resources and service resources. This resource sharing has technical and management factors. From technical aspect, the resource sharing domain can be implemented in the processing layer of the three-layer architecture. Hence, from technical point of view, the resource sharing

domain of the six-domain architecture can be treated as functionalities in the processing layer of the basic three-layer architecture. From management aspect, the resource sharing domain is beyond the coverage of the static three-layer architecture.

6. **Operation, management and control domain.** By definition, the operation, management and control domain has a strong connection with the service provision domain. In fact, operation, management and control cannot separate from service provision, because services need operation and management, and sometimes control as well. Even if the operation and management are conducted by an independent third party, the operation and management themselves are a kind of service. If supervision is treated as a special kind of service, then the service provision domain and the operation, management and control domain can be treated as a generalized service provision.

   Because the services provided by the operation, management and control domain is about the working situation of the whole IoT system, it can hardly be connected to any layer of the basic three-layer architecture. From this point of view, the operation, management and control domain cannot be mapped into any layer of the basic three-layer architecture.

The above analysis shows that, apart from the management factors, all the technical functionalities and services of the six-domain architecture can be realized by some layers of the basic three-layer architecture.

On the other hand, if the layers of the basic three-layer architecture are mapped into the layers of the six-domain architecture, it is easy to see from the above analysis that, the perception layer of the basic three-layer architecture corresponds to the object domain of the six-domain architecture, and can implement some functionalities of the perception and control domain and the service provision domain of the six-domain architecture; the network layer of the basic three-layer architecture corresponds to part of the functionalities in the service provision domain of the six-domain architecture; the processing domain of the basic three-layer architecture corresponds to the functionalities in the user domain, and partial functionalities in the service provision domain and resource sharing domain of the six-domain architecture.

In overall, the six-domain architecture is designed based on the functionalities of active and expandable IoT systems, while the basic three-layer architecture is designed based on the static IoT devices. These two architectures can be used to guide the construction of an IoT system in static status and in active status respectively.

2.8 Summary

There is no definition about IoT. However, the concept of IoT can be described by an architecture. Different IoT architectures have been proposed. The most commonly accepted architecture is the three-layer architecture, which says that an IoT system

is composed of a perception layer, a network layer, and a processing layer. The processing layer can further be split into a processing layer and an application layer, and hence a four-layer IoT architecture is formed. There are more than one three-layer IoT architecture, more than one four-layer IoT architecture. Only some typical IoT architectures are discusses in this chapter, and there are more IoT architectures in the public literatures that are not discussed here. Some relations between different IoT architectures are briefly described. It is shown that, different IoT architectures provide different angles to understand the components of an IoT system, and how an IoT system works.

These IoT architectures play an important role in guiding the construction of IoT systems, and are helpful in designing IoT security architectures. It is understood that, different IoT architecture may affect the way how an IoT system is constructed, e.g., from the point of view of physical components as a static IoT system, or from the point of view of functionalities and services by treating the IoT system as an alive system. Different IoT architectures affect the design of IoT security architectures.

References

1. M. Burhan, R.A. Rehman, B. Khan, B.-S. Kim, IoT elements, layered architectures and security issues: a comprehensive survey. Sensors **18**(9), 2796 (2018). https://doi.org/10.3390/s18092796
2. Z. He, K. Bai, D. Lin, C. Wu, Unification of identifiers in the Sea-Cloud system. Front. Comput. Sci. **12**(4), 749–762 (2018)
3. R. Khan, S.U. Khan, R. Zaheer, S. Khan, Future internet: the internet of things architecture, possible applications and key challenges, in *Proceedings of the 2012 10th International Conference on Frontiers of Information Technology (FIT), Islamabad* (2012), pp. 257–260
4. S. Madakam, R. Ramaswamy, S. Tripathi, Internet of things (IoT): a literature review. J. Comput. Commun. **3**, 164–173 (2015)
5. P. Sethi, S.R. Sarangi, Internet of things: architectures, protocols, and applications. J. Electr. Comput. Eng. **2017**, 9324035, 25pp. (2017)
6. M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, H.-Y. Du, Research on the architecture of internet of things, in *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE'10)*, vol. 5 (IEEE, Chengdu, 2010), pp. V5-484–V5-487
7. R. Zhang, R. Xue, D. Lin, Security architecture of sea-cloud (in Chinese). Sci. China Inf. Sci. **45**(6), 796–816 (2015)

Chapter 3
IoT Security Architecture

Based on the popular three-layer IoT architecture, which can naturally be treated as a four-layer architecture by separating the application from the processing layer, this chapter presents an IoT security architecture. The security architecture includes four layers, i.e., perception layer security, network layer security, processing layer security, and application security. Apart from the security requirements in each of the security layers, the proposed IoT security architecture also includes two supporting pillars, one pillar is trust and key management, which is the pre-condition to establish security functionalities of IoT systems, and the other pillar is operational supervision and security evaluation, which is the post-condition of IoT security services. The trust and key management supporting pillar should be made available before authentication and key agreement can be done reliably, and the operational supervision and security evaluation supporting pillar is usually provided by government or industry supervisors when the IoT systems are in operation. This pillar is to make sure that the required security functionalities and security services are provided complying with relevant specifications and standards.

3.1 Introduction

After the proposal of the concept of IoT systems, the content and features of IoT systems have gradually become clear. It is known that an IoT system has three typical functionalities: perception, transmission, and processing, hence they form the basic three-layer IoT architecture. It is anticipated that the techniques in IoT perception layer will become more and more smart, the techniques in IoT transmission layer will maintain the feature of being standardized, and the techniques in IoT processing layer will become more and more intelligent.

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_3

Different IoT architectures are proposed. Similar to the case of IoT architectures, there can be different IoT security architectures. Considering that the IoT security architecture is something to be built before the IoT system is constructed, the IoT security architectures can be built based on the four-layer IoT architecture as demonstrated in Fig. 2.3 in Chap. 2.

An IoT security architecture can be established based on an IoT architecture. However, from Chap. 2 it is known that, there are a number of different IoT architectures. In order to determine which IoT architecture is more appropriate to establish an IoT security architecture, we take the path of checking how the data is processed in an IoT application system [7].

First, consider the up-stream traffic flow. Some original sensor data is collected by sensors, then the data are transmitted to an IoT gate node which further transmits the data to a data center for further processing. The data center processes the data and provides services to end users via endpoint user devices, where the endpoint user devices may be small devices such as mobile phones or large devices such as personal computers or even large screens. The services that the processing layer provides to the endpoint user devices may be in the format of some kind of message to display, or some commands instructing the operation of the endpoint IoT devices. By treating the process of data consumption (e.g., when the data is like audio, video, or multimedia) the same as command execution, the up-stream traffic flow is almost always connected with a down-stream traffic flow.

It is obvious that, the down-stream traffic flow is just the opposite of the up-stream traffic flow. More precisely, a specific command (in the format of data) is sent by a user via an endpoint user device, then the command is sent to the processing center, which sends the command to the target IoT device for execution.

In the process of data transmission, the sensor data goes from the perception layer to the processing layer through the network layer. Within the perception layer, the data may be transmitted over some local networks, which are typically short distance wireless networks such as WiFi, Bluetooth, ZigBee, 433 MHz. The IoT application data usually go through an IoT gate, a fog computing platform, or an edge-computing platform, before it is transmitted over the network layer to the data center in the processing layer. When some data are sent from the processing center to the endpoint user devices, it forms a down-stream traffic flow.

From the traffic flow it can be seen that, security services are needed in every step of data transmission and processing. This means that, security services are required in the perception layer, in the network layer, and in the processing layer. Some characteristics of attacks and typical security approaches in different layers of IoT systems can be shown in Table 3.1.

It is noted that, when the data is used, many IoT applications may have some specific security requirements. For example, in a WIT120 (smart medical) system, user identity privacy is required, particularly with handling electronic medical records. It is noted that, specific security requirements such as privacy protection

Table 3.1 Some characteristics of attacks and security approaches in IoT systems

Physical layer	IoT devices damage	Physical protection
	Intrusion	Intrusion tolerance
	Illegal control	OT security techniques
	Energy consumption	Device dormancy
Network layer	DDoS attack	Distributed computing
	Man-in-the-middle attack	Authentication
	Traffic analysis	Data padding
Processing layer	Password guessing	Random password
	Illegal access	Intrusion detection
	Illegal operation	Access control
	Misbehavior	Security audit
	Privacy leakage/breach	Privacy protection

are not addressed properly in the three layers of IoT systems as general security requirements. Hence, it would be wise to group the security services for different IoT applications as the application layer security. This motivates that the IoT security architecture should treat the processing layer security and the application layer security separately. This yields a four-layer IoT security architecture.

3.2 A Layered IoT Security Architecture

In [3], a four-layer IoT architecture is proposed, as shown in Fig. 3.1. The layer names are similar to those in the four-layer IoT architecture as shown in Fig. 2.3, except that the processing layer is replaced by the support layer. However, from the content of each layer it can be seen that, this new four-layer architecture is actually a security architecture.

It can be seen that, the layers in the architecture of Fig. 3.1 are security services for the perception layer, the network layer, the processing layer, and the application layer of the architecture of Fig. 2.3 respectively. However, the discussion in the last section shows that this security architecture is insufficient to ensure security provision for IoT application systems, particularly from the viewpoint of IoT users, because the security services for each individual layers do not ensure the security provision for the whole IoT system.

To overcome the imperfection of the IoT security architecture as shown in Fig. 3.1, we hence propose a new IoT security architecture.

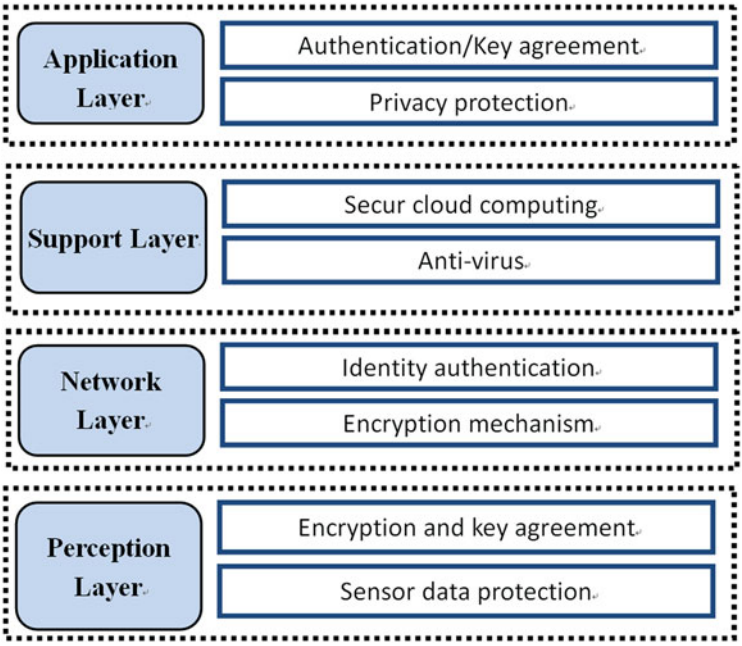

Fig. 3.1 A four-layer IoT security architecture

3.3 A "4+2" Structure of IoT Security Architecture

When considering IoT security services, the IoT system need to be considered as a whole system, this means that, even if the security mechanisms are provided in every IoT layer, the ultimate security services may not be provided properly. It is noted that there is an edge between every two IoT layers, hence security services penetrating through those edges may have vulnerabilities. On the other hand, supervision of security services and functions is another branch of security mechanism that an IoT security architecture should cover. Based on the above discussion, an IoT security architecture can be constructed which includes four security layers corresponding to the four layers of IoT architecture, and two vertical supporting pillars going through the former four security layers.

One supporting pillar is called *trust and key management (TKM)*, which provides security fundamentals to support the security services of other layers, i.e., the perception layer, the network layer, the processing layer, and the application layer. Another supporting pillar is defined as *operational supervision and security evaluation (OSSE)*, which is meant to provide supervision and test services for existing IoT systems. This security architecture has four layers and two supporting pillars, hence is called a *"4+2" structure of IoT security architecture*, and can be demonstrated in Fig. 3.2.

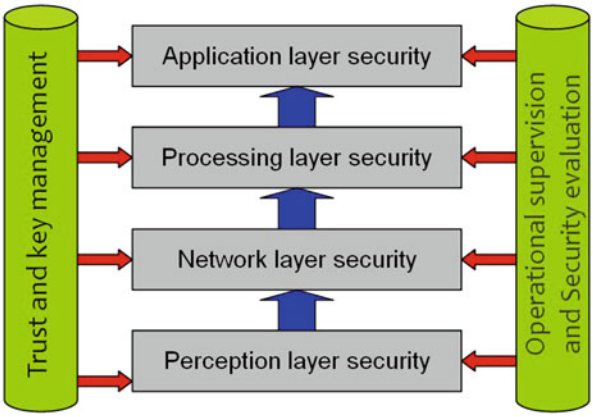

Fig. 3.2 A "4+2" structure of IoT security architecture

3.4 IoT Perception Layer Security Mechanisms

3.4.1 Perception Layer Security Requirements

The IoT perception layer is where the physical world is connected to the cyberspace. In the IoT perception layer, there are IoT devices, endpoint user devices, networks (e.g., short-distance wireless networks, for the connection of devices in the perception layer), and perhaps some processing platforms (e.g., edge computing and fog computing platforms). In order to provide security services for the perception layer, the following security requirements should be considered:

- **Physical security.** The physical IoT devices should be properly protected, including avoiding signal disturbance and interference (to ensure the proper working condition), avoiding electrical shielding and electromagnetic leakage, thwarting side-channel attacks, mitigating energy consumption attack and other network attacks.
- **Execution security.** Most of the IoT devices have an embedded operating system, hence should ensure the security of such an operating system, with security flaws patched in time.

  In addition, the execution security also includes security of application software. Software security is mainly about the damage caused by application software (e.g., malicious code) and misuse of software (e.g., illegal modification or breach of copyright).

  Another specific security requirement of IoT devices is the white-box implementation of cryptographic algorithms [2], which is an implementation technique aiming at reducing the chances of attacks from unauthorized end users. This is also a special kind of execution security.

- **Data security.** The data confidentiality and data integrity need to be provided properly during data transmission and data storage. In addition, backup and recovery of important data (e.g., configuration data) are also important content of data security in the IoT perception layer.
- **Network security.** Data transmission is conducted over networks, hence network security is closely related to data security. The above data security is not enough to ensure secure data transmission, there may exist data forgery, replay attack, network intrusion, and other kind of attacks conducted over the networks. Hence network security protection is necessary in the perception of many IoT systems.
- **Behavior security.** Tradition security techniques mainly consider how to protect the target from being attacked by others. The behavior security is about eliminating the IoT devices not to be involved in attacking other devices. This is a special security requirement specific to IoT devices.

The above security requirements can be further split into more specific requirements. Particularly, the security threats and some corresponding methods to thwart/mitigate the security threats for IoT devices, up-stream data, and down-stream data can be made more specific.

3.4.2 Security Threats and Methods of Protection for IoT Devices

In the perception layer, different IoT devices work differently. Sensors capture data of the environment, actuators execute commands from down-stream data, and IoT gate nodes and other network devices transmit data with or without data processing. Hence, IoT devices can be the targets of network attacks.

Security requirements for IoT devices should take into consideration the devices themselves as well as the data that they process. Hence security threats on IoT devices may include the following cases:

1. Energy consumption attack and denial of service (DoS) attack. The DoS attack is to make a victim IoT device not be able to respond network connections temporarily, while the energy consumption attack is to make a victim IoT device to shut down completely by exhausting its battery power. The energy consumption attack is more specific to IoT devices using batteries to supply power.

   There does not seem to have efficient method to thwart DoS attack. However, with respect to the energy consumption attack, an effective method is to let the IoT devices to sleep, i.e., for those devices that network connection is not always needed, they are put into sleep (or simply close the network connection) is an effective method to mitigate energy consumption attack.
2. IoT devices may be physically captured and being analyzed in a laboratory. The laboratory analysis is often more powerful than network attacks, which has more chances of being able to get some confidential information inside the devices such as their encryption keys.

Security protection against laboratory analysis can be done by techniques such as hardware packaging and resistance against hardware unwrapping.

3. IoT devices may have been controlled by intruders. Particularly when such devices are connected to the Internet directly, such as gate nodes and webcams.

   Depending on the effect of malicious intrusion, security requirements include intrusion detection [6], intrusion tolerance [8], identity authentication, and other cooperative mechanisms. Intrusion detection only suits the devices with large computing resources, intrusion tolerance targets at Industrial IoT where the purpose of intrusion is illegal control of other endpoint devices, and identity authentication is to let other devices to be able to detect the illegal communication from the intruded devices.

3.4.3 Security Threats and Methods of Security Protection for Up-Stream Data

Up-stream data is mainly about environmental data, including sensor data, location information, meter reading, surveillance video, RFID tag information, and other information about IoT devices. Security threats and methods of protection for up-stream data include the following:

1. Eavesdropping and unauthorized access. During the process of data transmission, eavesdropping is a common security threat for confidential data. If the data is stored somewhere, then unauthorized access is a common security threat. The purpose of eavesdropping and unauthorized access is to get the content of the data.

   Encryption is a common method to provide data content protection. An encryption algorithm transforms plaintexts into ciphertexts, and only the ciphertexts are transmitted. Unauthorized users can hardly recover the plaintext even if they can capture the ciphertexts. There are a variety of encryption algorithms, many of them have been widely used.
2. Data-forgery. Attackers may try to create some fake data pretending to be valid data, this is data-forgery attack. If the recipient cannot differentiate the authorized data and forged data, then data-forgery attack can be successful.

   The data-forgery attack can be prevented in many ways. Specific methods to prevent data-forgery attack include data source authentication, data integrity protection, or a proper use of encryption.
3. Traffic flow analysis. If the transmitted data is encrypted, then eavesdroppers cannot get the corresponding plaintext of the captured ciphertexts. However, since traditional encryption algorithms usually map messages into ciphertexts of similar size, i.e., a large-size message is encrypted into a large-size ciphertext, and a small-size message is encrypted into a small-size ciphertext. By checking the size of the ciphertext, attackers may be able to get some information. For

example, short ciphertext of some household's water consumption indicates that the household is likely not occupied.

A method to thwart traffic flow analysis is to balance the size of ciphertexts, making the ciphertexts to be of similar size for plaintexts of different sizes. A common method to do this is to use data padding, i.e., if the largest allowable size of plaintext is n bytes, then by padding every plaintext into n bytes, their ciphertexts can be in the same size, and traffic flow analysis reveals very little information.

Other attacks such as illegal modification of data and data replay-attacks are also security threats. However, since the implementation of such attacks need sophisticated tools, practical protection against such attacks has not yet become popular.

3.4.4 Security Threats and Methods of Security Protection for Down-Stream Data

Down-stream data is mainly about commands, instructions, warnings, and messages for endpoint devices to display. Security threats and methods of protection for down-stream data include the following:

1. Data-forgery. If data-forgery can be made, then it would be a severe security threat, because the forged data can be a malicious command or instruction, which would cause wrong operation and severe damage.

   There are mature techniques to prevent data-forgery. One of effective method is to use authentication technique to verify the data source, and use data integrity service to prevent illegal modification as well as forgery.
2. Illegal modification. Illegal modification of data is similar to data-forgery, except with some valid data available to manipulate.

   Message authentication code (MAC) is a mature and effective method to prevent illegal modification of data, the mechanism is that modified data can be detected by the receiver. The chances of successful cheating is negligibly small.
3. Replay attack. If a message is captured by and attacker and is replayed in a later time, if the message is accepted by the receiver, then the receiver may process the data as with valid data. If the data is a command, then executing a command in a wrong time can cause severe problems.

   Replay attack can hardly be prevented by usual encryption and MAC algorithms, because the captured data is valid, can be correctly decrypted and can pass integrity verification. Effective methods to prevent replay attack is to add a timestamp, or a counter (e.g., if the devices do not have a clock available), such added message is called *freshness tag*. By checking the freshness tag, the receiver is able to tell whether the data is fresh or outdated.

Note that an active replay attacker may modify the freshness tag to make it valid, this kind of attack is called modified replay attack. Prevention of modified replay attack is to put the freshness tag into security protection, e.g., being encrypted or with integrity protection.

It is noted that data confidentiality as a very fundamental security requirement may not be very crucial here, because in many IoT applications, the content of the down-stream data does not need confidentiality. For example, the command to operate a switch is not a confidential message, but the integrity of such a command is crucial.

3.5 How to Ensure IoT Perception Layer Security

When setting up the security protection for the IoT perception layer, the following factors need to be considered.

1. **Determine the security requirements.**
   Different applications have different security requirements. Some applications need message confidentiality service, while some others may need message integrity service. Most applications over the Internet need authentication service. In the case of command data that instructs the operation of IoT devices, the message freshness service is also important.

   If an IoT application system requires message confidentiality service, it needs to consider to what strength confidentiality service is needed. For example, only symmetric key encryption is needed, or public key encryption is also needed; whether session keys are required, or data encryption simply uses the seed key.

   If an IoT application system requires message integrity service, since a message transmitted over the network usually includes different parts, for example, the identities or the addresses of the sender and the intended receiver, a timestamp (or a counter), an initial vector, and the application data, hence it needs to clarify what parts of the message need integrity protection.

   If an IoT application system requires authentication service, it needs to consider how to balance the convenience and the security level. It is also important to consider whether multi-factor authentication is needed; whether unified authentication mechanism is needed, i.e., one authentication can be used to access different resources.

   The security requirement analysis is critical to choose the appropriate security techniques.
2. **Choose security techniques that can provide security services meeting the security requirements.**
   Once the security requirements are made clear, the next step is to choose what techniques are used to provide the required security services.

   If an IoT application system requires symmetric key encryption for message confidentiality service, then an appropriate encryption algorithm should be

chosen. There are a number of standard encryption algorithms for commercial use, some of them have been properly implemented in hardware or software, using such an algorithm may be a convenient choice. The same problem happens with the requirement of public key encryption, the requirement of message integrity service, and the requirement of authentication service by using digital signatures.

It is noted that most of the IoT application systems require more than one type of security service, then the choices of the security techniques need to be considered as a whole. In some cases, one security technique may provide more than one security services. For example, a digital signature scheme may provide identity authentication service, non-repudiation service, and message integrity service. The example in Sect. 12.6 shows that, proper use of an encryption algorithm may provide multiple security services, including message confidentiality, message integrity, source authentication, and data freshness service.

While choosing proper cryptographic algorithms, a proper method for trust establishment and key management is also necessary. The establishment of trust is often made off-line, conventionally either by assigning a pre-defined secret key (as a long-term key), or a public key certificate (if a PKI system is available). There is no standard way of choosing those techniques, however, considering that many IoT devices have constrained resources, lightweight security techniques are favourable in IoT applications.

3. **Determine the security design.**
In choosing security techniques, it is easy to simply adopt those that have wide applications in traditional information systems. However, this principle is inappropriate for IoT applications.

When designing security mechanisms and choosing the corresponding security techniques, the following rules/principles are advised.

(a) Try to avoid public key algorithms, if possible. This is because the public key algorithms usually need large amount of hardware and software resources, and the public key management may involve considerable amount of overhead cost on the resources. This is due to the consideration on the lightweight feature of security techniques.
(b) Try to avoid challenge-response mechanism in authentication protocols, because message exchange consumes relatively large amount of energy, usually much more than computation such as encrypting the same size message. It would be ideal to use one-way message transmission to fulfill multiple security services.
(c) Try to avoid using random number generators or even pseudo-random number generators, because such generators often require extra hardware and/or software resources.
(d) If a message conveys a command, then data freshness service is important, particularly when the communication data can be accessed publicly. This is the case when the transmission is over the Internet or via wireless networks. If the application has a system clock available, then try to use the timestamp

to offer data freshness service; otherwise, use a counter as the message freshness tag.

(e) Try to set clock synchronization together with application data transmission. By doing this, there is no need for separate clock synchronization.

3.6 IoT Network Layer Security Mechanisms

The IoT network layer security should have overall consideration of cost, convenience, and the security requirements. It is obvious that the security services provided by the network layer are from network operators, such as an Internet service provider or a mobile network operator. With the fast development of wireless networks, particularly the 3G/4G/5G mobile communication networks, the use of such networks in IoT applications becomes very convenient, and with affordable price. Security mechanisms in the network layer, regardless whether they are from the Internet or from mobile communication networks, are usually established by international organizations. For users and designers of an IoT application system, the security mechanisms of the network layer is simply to choose the security services offered by the networks.

Apart from the Internet or mobile communication networks, a new kind of networks, called *low power wide area networks (LPWANs)*, are invented. Typical examples of LPWANs include narrow-band IoT (NB-IOT), LoRa, and SigFox, they have been commercially used. The security mechanisms of those LPWANs have not been deeply studied as those of other traditional networks.

3.7 Typical Security Techniques for IoT Network Layer

As mentioned above, the Internet and the mobile communication networks compose the network layer for most IoT applications. There are a number of standard security protocols used by the Internet, they can be categorized into different layers of the computer network architecture. Some typical security protocols and their brief description are given in Table 3.2.

Security techniques in mobile communication networks typically include two phases: the authentication and key agreement (AKA) phase and secure data communication phase. The AKA phase is to complete the identity authentication of users requesting a connection to the network, and the identity authentication of a service provider, and in the meantime, to set up common keys between the user and the service provider. The secure data communication phase is to provide transmit data with confidentiality and integrity protection during wireless transmission.

In a GSM network, authentication is done between a user equipment (UE) and an authentication center (AUC). There is a visitor location register (VLR) in between the UE and the AUC, which is involved in the authentication process. When an UE requests to connect to the network, the AUC finds the secret key shared with

Table 3.2 Some typical network security protocols

Layer	Security protocol	Description
IP layer	IPSec	IPsec has some sub protocols: the Encapsulating Security Payload (ESP) protocol provides a method for encrypting data in IP packets, and the Authentication Header (AH) protocol provides a method for digitally signing IP packets. The Internet Key Exchange (IKE) protocol is used to manage the cryptographic keys used by hosts for IPsec. IPSec has two modes: Tunnel mode, providing point-to-point security, is usually used between secured network gateways; Transport mode, providing end-to-end security, is used for secure communication between two individual machines.
TCP layer	SSL/TLS	Secure sockets layer (SSL) and its improved version—Transport layer security (TLS) are designed to provide transport layer security (in the TCP/IP model). They form a separate security layer, using cipher suits to provide data confidentiality and integrity services, as well as host authentication service.
Application layer	SET	Secure electronic transaction (SET) protocol was designed for electronic payment security. It provides card holder authentication and transaction data confidentiality services.
	S/MIME	S/MIME is security enhanced Multipurpose Internet Mail Extensions protocol, it was designed by RSA cooperation to provide message confidentiality, integrity, and source authentication. Another secure protocol for emails is Pretty Good Privacy (PGP), which has its own security mechanisms and techniques.
	HTTPS	HTTPS is security enhanced hypertext transfer protocol, it is based on SSL/TLS to provide security from the transport layer connection.

UE, computes a number of authentication vectors and forward the authentication vectors to the VLR, the VLR selects one authentication vector, sends the random number in the authentication vector as a challenge to UE, UE can compute a response to the challenge using its permanent secret key, and the VLR can compare whether the response from the UE is the same as the expected response in the authentication vector. If they match, then authentication is complete. The UE can then compute a session key using its permanent secret key. Since the session key is in the authentication vector, the VLR can establish a secure channel with the UE using the session key. More detailed description of the authentication process can be found in Chap. 7. Since the authentication process is immediately followed by a session key agreement, the authentication process is called *authentication and key agreement (AKA)*.

The GSM system only provides the mechanism for the network to authenticate mobile users. Due to lack of network authentication, security fraud is possible. One type of such fraud is the appearance of fake mobile stations, which forge large amount of fake SMS services luring victims to malicious network links and phone numbers operated by the telecom fraud teams.

Noticing the security drawbacks, the 3G networks have greatly improved the security level, including enabling users to authenticate the networks. The 3G networks also improved encryption algorithm, and provide message integrity service. However, the mechanism of AKA is much like that in GSM, in the sense of using the challenge-response mechanism for authentication, and the use of authentication vectors. The long-term-evolution (LTE) networks, with some new versions of LTE being known as 4G networks, have many changes over 3G. However, the challenge-response mechanism of authentication and the use of authentication vectors remain the same.

Apart from the Internet and mobile communications networks, the low power wide area networks (LPWAN) have also been commercialized and widely used in IoT applications.

3.8 IoT Processing Layer Security Mechanisms

The IoT processing layer is mainly about platforms providing services for centralized data processing, e.g., in a cloud computing platform. Sometimes a normal workstation may act as the processing layer in a small-scale IoT application. The IoT processing layer includes the data processing platform itself and the services it provides. With respect to the security of the platform, the following security problems need to be considered:

1. Operating system security for the platform, including user management, access control, security audit, intrusion prevention and intrusion detection.
2. Data security, including data backup and data recovery mechanisms. This is in case when the system does not function properly, for example, when the system is damaged by attackers.
3. Hardware backup mechanism. This is in case when some part of hardware is broken, how to reduce the affect when such hardware failure occurs.
4. Concurrent computing, which reflects the capability to handle large amount of simultaneous visits. This mechanism is to ensure that the platform is able to serve certain amount of users properly. It should also include the mechanism to dynamically increase the capability to handle increased peak visit.
5. Technique to handle DoS attacks and DDoS attacks. Although such attacks are difficult to avoid, some mechanism to mitigate the effect is possible and helpful.

With respect to security services in the processing layer, the above list can be further extended, because the security services are variable. Some typical such services are as follows.

1. Data confidentiality mechanism. This is to ensure that user data are encrypted when being stored in the processing platform (e.g., the cloud).
2. Data integrity mechanism. This is to ensure that user data are not illegally modified during their storage in the processing platform. In addition, some mechanism to allow users to efficiently check the integrity of their data is also a practical security requirement.
3. Data query service. Data query is a common service provided by many database centers. However, for encrypted data, data query becomes a challenging problem. The techniques enable data query on encrypted database is called searchable encryption.
4. Cloud-based outsourcing computation. The processing layer can provide computing services for some devices with limited computation capability. While outsourcing computation of normal computation is a trivial task, outsourcing of secure computation is a challenging problem, because the process of secure computation involves secret information, which is not supposed to be known by the processing platform.

Apart from the above mentioned services, many traditional applications become cloud computing services. Typical such services include *Infrastructure-as-a-Service, (IAAS)*, *Platform-as-a-Service (PaaS)*, *Software-as-a-Service (SaaS)*, *Computation-as-a-Service (CaaS)*, *Data-as-a-Service (DaaS)*, *Security-as-a-Service (SecaaS)*, *Testing-as-a-Service (TaaS)*. More such services are invented and added to the list, such as *Evaluation-as-a-Service (EaaS)*, *Mobility-as-a-Service (MaaS)*, and it tends to be everything-as-a-service.

With respect to security related services, some mechanism to reduce the risk of internal attack should be designed, because many network attacks and information leaking events were due to internal attacks.

It is known that most cloud platforms use the technology of distributed computing, which can reduce the chances for the system to shut down completely. With the development of cloud computing, many new security services and corresponding techniques may emerge.

3.9 IoT Application Layer Security Mechanisms

The application layer security problems come from the specific security events from different industry and IoT applications. Although the application layer security is mostly achieved via the data processing platforms (i.e., the processing layer), the services provided by the processing layer often cover multiple applications, while the application layer security is more specific in different IoT applications.

Some typical application layer security services include the following:

1. Privacy protection. Although the privacy protection is not a security requirement in all the IoT applications, it is an important security measure in some IoT applications.

There are two types of privacy in network environment, i.e., identity privacy [1] and location privacy [4]. The purpose of identity privacy protection is to protect illegal leakage of user identities. For example, in a smart medical system (or WIT120), electronic medical records (EMRs) are often used for medical science research and pandemic analysis, in such cases, the identities and any information that can be used to track the patents' identities in the EMRs should not be revealed, this is where identity privacy protection is needed.

The location privacy service is useful in allocating and tracking rare animals for scientific research, however, the location information as a location privacy information should be protected from animal hunters.

2. The whole security of an IoT application system. Security protection is not the result of a collection of techniques, it should be designed as a whole. The security of the whole IoT system is subject to the point with the weakest security protection. By saying weak or strong, it is an evaluation based on the security threats and security techniques used.

   The whole security of an IoT application system is determined by the security policies, security design based on the policies, security techniques, and security management, which includes how trust is established and how the symmetric keys and public keys are managed, how the security functionalities are tested, how the operation and maintenance are supervised, etc..
3. The security of user mobile devices. Security risks of user mobile devices include the process of operation being peeped, password being stolen, malicious codes and malicious links being activated, and the devices being stolen. If a user mobile device is authorized to send commands to instruct the operation of important IoT devices, then a stolen device, or a forged device, can cause unpredictable damage to the IoT application system.

Given the above notification, some typical security techniques for IoT application layer include the following:

1. While IoT techniques provide a lot of convenience to our life, people may worry about whether their private information is leaked. This is identity privacy problem. Identity privacy is associated with the information that can be used to determine the identity of a specific person (e.g., a user or a customer), including name, mobile phone numbers, detailed living address, etc.. Techniques for identity privacy include encrypting the private information, deleting the private information, replacing the private information with random strings, or some techniques of fuzzification [5].
2. Location privacy can be provided by encrypting the location information, so that only authorized users are able to get the information. Alternatively, location privacy can be protected by using a large number of fake location messages that can make confusion.
3. When both the anonymity and authenticity services are needed, anonymous authentication techniques become useful.

3.10 The Establishment of Trust and Key Management in IoT Systems

The proposed IoT security architecture has a vertical supporting pillar called trust and key management (TKM), it goes through the perception layer, the network layer, the processing layer, and the application layer. This means that the trust and key management services are needed in all those layers.

The necessity for the trust and key management is that, the security services provided by the conception layer, the network layer, the processing layer, and the application layer, do not ensure the security of the IoT application system as a whole. The reason is that, the concept of security must be based on a security model. The security of different IoT layers is based on different models. More precisely, the security of perception layer is based on the assumption that attackers are beyond the communication parties. However, when the communication parties are treated as possible attackers (or opponents, because they are not in the same benefit community of the IoT application system), then the security anticipation is broken. For example, the network layer security based on mobile communication networks can be ensured, because mobile communication networks use international standard security techniques to provide data confidentiality, data integrity (since the 3G network), and entity (i.e., data source) authentication. However, inside the mobile communication provider, the content of communication is available in plaintext form. From the point of view of an IoT application system, the mobile communication providers are not part of authorized users who are not supposed to be able to access the content of their data. This security flaw comes from the layered security mechanism.

The purpose of trust is to provide a starting point for authentication and key management. Based on the trust assumption, reliable secure communications can be established, and the security services for an IoT application system as a whole can be made possible.

The TKM pillar provides such a mechanism to enable security services going through different layers of IoT. For example, based on the established trust, IoT devices can establish common keys, they can be used for data encryption and data integrity protection. The data confidentiality and data integrity services remain when the message goes through the perception layer, the network layer, and the processing layer, hence can provide end-to-end security.

While the TMK pillar is meant to provide security for the whole IoT application system, the layered security mechanism is still valuable.

3.11 Operational Supervision and Security Evaluation

Operational supervision and security evaluation (OSSE) is another vertical supporting pillar in the IoT security architecture as shown in Fig. 3.2. Like the trust and key management pillar, the security evaluation and operational supervision also

goes through all the other layers, i.e., the perception layer, the network layer, the processing layer, and the application layer. This means that the security evaluation and operational supervision services cover all those layers.

Operational supervision is compulsory for most industry activities. For information security related services, in order for the supervision to be effective, security evaluation is conducted regularly. This is to ensure that the industry has proper security protection.

Security evaluation is often based on security test. If an IoT system passes the security test, then it is believed that the tested system has proper security provision. Due to that an IoT system is composed of so many components, the task of IoT security test can focus on the perception layer and the whole IoT system, because the security evaluation on the network layer can be done using many traditional methods, and the evaluation on the processing layer is something to do about cloud computing security, which is usually treated as an area separated from IoT.

While the TMK pillar is to support an IoT security in its construction process, the OSSE pillar works after the IoT security system has been constructed and is in working status.

3.12 Summary

This chapter presents a security architecture of IoT. The security architecture is based on the three-layer IoT architecture as shown in Fig. 2.1 or equivalently the four-layer IoT architecture as shown in Fig. 2.3. As the cases of IoT architectures, there can be different IoT security architectures.

The presented IoT security architecture is composed of four layers and two supporting pillars. The four security layers are corresponding to the layers of the four-layer IoT architecture as shown in Fig. 2.3, while the two supporting pillars provide supplementary support to the security services of the four layers. The rational about the necessity about each part of the IoT security architecture is briefly described, and will be further discussed in other chapters.

It should be noted that the rough description about the IoT security architecture in this chapter is not enough to instruct practical applications. Further study on the IoT security architectures and security techniques is necessary. The following questions should be considered when studying the IoT security architectures:

1. **Necessity:** Why each part of the architecture is necessary, what might be the effect if any part is missing?
2. **Completeness:** Is the security architecture (existing one or newly proposed) able to provide proper security protection for most IoT application systems? What security services can be easily neglected if the architecture is used to guide security implementation?
3. **practicability:** Are there relevant techniques available to provide necessary security services as instructed in the architecture?

It is noted that the IoT security architecture described in this chapter is unable to answer all the above questions, further study on this topic is needed.

References

1. K.C. Chang, R.N. Zaeem, K.S. Barber, Enhancing and evaluating identity privacy and authentication strength by utilizing the identity ecosystem, UTCID Report #19-03, The University of Texas (2019)
2. S. Chow, P. Eisen, H. Johnson, P.C. Van Oorschot, White-box cryptography and an AES implementation, in *International Workshop on Selected Areas in Cryptography (SAC)* (Springer, Berlin, 2002), pp. 250–270
3. D. Darwish, Improved layered architecture for internet of things. Int. J. Comput. Acad. Res. **4**(4), 214–223 (2015)
4. M.J. Gajjar, Sensor security and location privacy, in *Mobile Sensors and Context-Aware Computing*, chap. 9 (Elsevier, Amsterdam 2015), pp. 223–265. ISBN: 978-0-12-801660-2
5. E. Kayacan, M.A. Khanesar, Fundamentals of type-1 fuzzy logic theory, in *Fuzzy Neural Networks for Real Time Control Applications (Concepts, Modeling and Algorithms for Fast Learning)*, chap. 2 (Elsevier, Amsterdam, 2016)
6. H. Lee, Framework and development of fault detection classification using IoT device and cloud environment. J. Manuf. Syst. **43**(2), 257–270 (2017)
7. G. Veneri, A. Capasso, Industrial data flow and devices, in *Hands-on Industrial Internet of Things*, chap. 2 (Packt Publishing, Birmingham, 2018)
8. W. Zhao, Intrusion tolerance techniques, in *Encyclopedia of Information Science and Technology*, 4th edn. (IGI Global, Hershey, 2018), pp. 4927–4936

Chapter 4
Fundamentals of Cryptography

The basic information security services are known to include confidentiality, integrity and availability, which are know as the CIA triad. This chapter introduces some basic concepts and techniques in cryptography, particularly some typical cryptographic methods to the CIA triad, i.e., encryption algorithms, message authentication codes, and digital signature algorithms. Encryption algorithms can further be categorized into symmetric key encryption algorithms and asymmetric key encryption algorithms, where the asymmetric key encryption algorithms are also called public key encryption algorithms. Some fundamental authentication protocols are introduced, which described how the authenticity of a remote party can be validated, e.g., how to verify an identity that a remote party claims to be is genuine.

4.1 Introduction

There are a few similar terms frequently used in this chapter, they are message, data and information. For the convenience of description, we briefly clarify their relations. Message is a piece of digital signal that can represent any kind of things. Message is a term commonly used in communications, meaning that the message is a representation of something in the communications; Data is a collection of digital strings (binary strings in particular) that can be processed and stored by computers, it is a term commonly used in computers, meaning that data is something associated with storage; Information is the content of message or data that can be understood by human, meaning that information is something associated with applications. Our description may alternatively use these three terms in different scenario, and hope that it would not cause confusion from the context.

Cryptography is a subject about techniques for information security. However, there is no appropriate definition about what information security is. Intuitively,

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_4

information security is to ensure the original purpose of the message and data, so that they cannot be misused.

In order to understand what the information security is about, we need to know what an opponent can do on a message during its storage and transmission. There are a number of possibilities that an opponent can do, some typical security threats are as follows:

1. Illegal access. This is a basic threat, and the purpose of obtaining the message is against the willingness of the message owner/creator.
2. Illegal modification. This threat is not just to destroy the content of the message, but to let the illegal modification convey a wrong message, intending to mislead the message receiver/owner.
3. Impersonation. Attackers may impersonate someone to send unauthorized messages.
4. DoS and DDoS attacks. Attackers may lunch denial of service (DoS) and distributed denial of service (DDoS) attacks.

The above basic security threats have corresponding primary security measures, they are confidentiality, integrity, and availability, known as CIA triad. The meaning of CIA is as follows:

- **Confidentiality** is about the protection of the content of messages, ensuring that the content of the message cannot be obtained by unauthorized users;
- **Integrity** is about the detection on illegal modification of messages, ensuring that the format of the message has not been illegally modified;
- **Availability** is as its name tells, i.e., authorized users can get the information and services that they are entitled to, and unauthorized users cannot get the information and services that they are not entitled to. While confidentiality and integrity are about messages, the availability is related to the messages as well as the platforms providing services over the networks.

The confidentiality measure is to prevent illegal access of the content of messages, the integrity measure is to detect illegal modification of messages, and the availability measure is to minimize the impersonation attacks and DoS/DDoS attacks.

Apart from the CIA, there are other extended security measures, which are targeted at different attacks and security threats. The following are some well-known extended security measures.

- **Non-repudiation** is about the provision of convincing evidence of someone having done something. This evidence can be used by a third part or judiciary to judge whether one's claim is a repudiation, hence is known as "non-repudiation" service.
- **Controllability** is about the total control over the operation of a system, so that malicious operations become visible and the chances of such operation to succeed can be minimized.

- **Freshness** is about whether a message is new and valid. It has two aspects: (1) The message is not a historic message; (2) The message reaches the receiver in a valid time period, if such time slot has been set. The purpose of freshness is to prevent re-play attack.

Apart from the above fundamental security measures, more security threats can be found and more security measures can be established. Chapter 12 lists over 60 security measures, mainly focused on the perception layer of IoT architecture.

This chapter introduces some primary security techniques that can be used in IoT systems.

4.2 Cryptographic Algorithms and Their Security Services

Practically, it is not possible to prevent an opponent to obtain a message that is transmitted over networks, particularly when it is transmitted over wireless networks. However, it is possible to protect the content of the message, so that the opponent cannot get any useful information carried by the message, while the content of the message can be obtained by the authorized receiver. The most common technique to protect the content of the messages is data encryption, and the security services provided by data encryption is called *confidentiality*, i.e., the content of the data.

Although data encryption can protect the content of the message from unauthorized access, an opponent may try to intercept the message, modify it and then send it to the original receiver. If the modified message is accepted by the receiver, since it becomes a wrong message, it may cause some problems. This kind of attack is called illegal modification attack, and the problem is called message integrity problem. A common technique to protect message integrity is message authentication code (MAC). The security service provided by an MAC is called *message integrity*, meaning that any illegal modification of the message can be detected.

Encryption algorithms and message authentication codes are two fundamental cryptographic methods. Another fundamental cryptographic method is called *digital signature* which is based on public key cryptography, which will be introduced later.

4.2.1 Data Encryption

Data encryption is a method to transform a message into a different form, where an encryption key determines how the transformation works. By Shannon's theory, an encryption algorithm should include two kinds of transformations, i.e., confusion and diffusion. The confusion transformation mixes the encryption key and the original message, and the diffusion transformation converts the message into a form that looks like a random number. The original message is called *plaintext*

from which the content can easily be extracted, while the transformed message is called *ciphertext*. The ciphertext can be transformed back to the original plaintext by the corresponding decryption algorithm and the correct decryption key. In some encryptions, the size of the plaintext and that of the ciphertext are fixed. Let the size of the plaintexts be n and that of the ciphertexts be l, then the set of all the possible plaintexts, i.e., the set $\{0, 1\}^n$, is called *the plaintext space*, and the set of all the possible ciphertexts is called *the ciphertext space*, which may be the set $\{0, 1\}^l$ itself, or a subset of it. It is obvious that the inequality $n \leq l$ must hold, otherwise, some plaintext cannot be recovered after encryption.

There are two types of encryption algorithms, they are called *symmetric key encryption algorithms* (or briefly *symmetric encryption algorithms*) and *asymmetric key encryption algorithms* (or briefly *asymmetric encryption algorithms*). For a symmetric key encryption algorithm, the corresponding decryption algorithm uses the same key, or the decryption key can easily be obtained given the encryption key. In most of the commercial symmetric encryption algorithms, their corresponding decryption algorithms use the same key as what is used for encryption.

4.2.2 Symmetric Key Encryption Algorithms

Denote by E_k an encryption algorithm using the encryption key k, and D_k the corresponding decryption algorithm using the decryption key k. Then the process of encryption and decryption can be represented as

$$c = E_k(m) \tag{4.1}$$

$$m = D_k(c) \tag{4.2}$$

where m is a plaintext, and c is the corresponding ciphertext. The processes of encryption and decryption can be depicted by Fig. 4.1.

There are different types of symmetric key ciphers with different ways to process the encryption and decryption. One type is called *stream cipher* and the other type is called *block cipher*. A stream cipher encrypts messages bit-by-bit, or byte-by-byte, with the encryption process being basically the exclusive-OR (i.e., XOR) operation of the plaintext and a key stream, where the key stream is generated by a pseudo-random number generator (PRNG) with the encryption key as the initial state (or

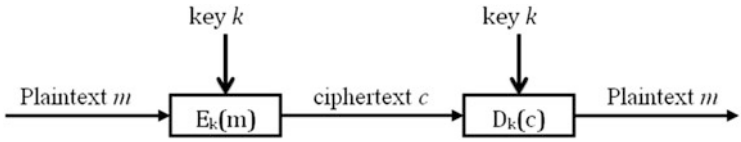

Fig. 4.1 The process of symmetric key encryption and decryption

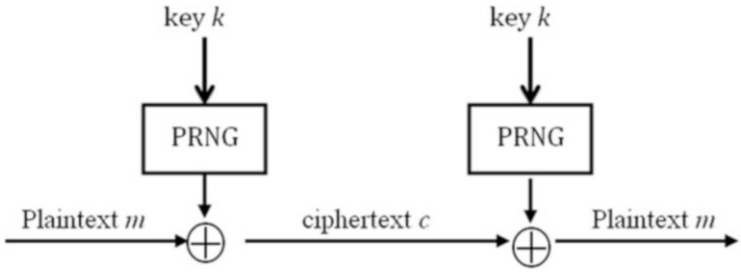

Fig. 4.2 The encryption and decryption process of a stream cipher

part of the initial state). This means that the length of the key stream should be no less than that of the plaintext. In fact, since a PRNG can generate the key stream as long as needed, a stream cipher does not have restriction on the length of the plaintext to be encrypted. Since the operation of the encryption process of a stream is the XOR operation, the decryption is also the XOR operation, but the inputs are the ciphertext (as a stream of bits/bytes) and the key stream. The encryption and decryption processes of a stream cipher can be depicted by Fig. 4.2.

Different from stream cipher, a block cipher algorithm encrypts messages a block at a time, i.e., a plaintext block is encrypted into a block of ciphertext, and the encryption processes plaintext messages in blocks. The length of the message block is called *block length*, which is usually 64 bits or 128 bits in the well-known block cipher algorithms. It is obvious that, in order for the original message m to be recovered by the decryption algorithm, the size of the message m must not exceed the block length. If the size of the plaintext message m is less than the block size, then some redundant message is added into the plaintext m making it to be m' with size being exactly the same as the block length. The process of adding the redundant message is called *padding*, and the redundant message is removed after the ciphertext is decrypted.

It is easy to see the identity $D_k(E_k(m)) = m$ holds for any message in the plaintext space. However, the equation $E_k(D)k(c)) = c$ may not hold for some message c, because an arbitrary message c may not be a valid ciphertext under the encryption key k. If the size of ciphertexts and that of the plaintexts have the same length, then it can be proven that, for any c in the ciphertext space, the equation $E_k(D_k(c)) = c$ always holds as an identity.

Typical symmetric block ciphers include data encryption standard (DES) [1], advanced encryption standard (AES) [2], international data encryption algorithm (IDEA) [3], RC2 [4], and SM4 [5], and typical stream ciphers include RC4 [6], the A5 algorithm [7] and ZUC algorithm [8] that are used in mobile communications, and the candidates for eStream program [9]. It is noted that some modes of operation of a block cipher can have the same effect as a stream cipher, which encrypts plaintexts byte-by-byte. For example, in 3G mobile communication systems, the block cipher KASUMI is used to form a key stream generator $F8$, which is used for message encryption in the way similar to a stream cipher.

4.2.3 Structures of Symmetric Block Ciphers

Every symmetric encryption algorithm has its own design. However, most symmetric block ciphers are based on two fundamental structures, they are Feistel structure and substitution-permutation (S-P) structure.

The Feistel structure was designed by Horst Feistel. The specific feature of Feistel structure is to encrypt half of the message in every round of encryption process, and one complete encryption process includes a number of encryption rounds. The most successful cipher based on Feistel structure is DES [1]. Figure 4.3 depicts the encryption process of DES, which is composed of 16 encryption rounds.

The round function $F(K, R)$ takes the round encryption key K and the right half of message block from the previous round, and produces a 32-bit string, which is then XOR'd with the left half of message block from the previous round to output the right half of message block of the current round, while the right half of message block from the previous round is copied as the left half of message block of the current round. The design of the round function $F(K, R)$ is quite sophisticated, and the generation of the round keys from the original encryption is an independent process.

It is interesting to note that, the decryption process of a Feistel structured encryption (e.g., DES) can be designed to be the same as the encryption process except that the round keys are used in reverse order. For example, if DES encryption algorithm use the round keys $K_1, K_2, \ldots, K_{16}$. By putting them in reverse as $K_{16}, K_{15}, \ldots, K_1$, then the encryption algorithm becomes the corresponding decryption. This feature can save hardware resources when both the encryption algorithm and the decryption algorithm are needed.

The S-P structure is to use the substitution and permutation as two fundamental operations in the encryption process. The S-P structured encryption algorithms also have a number of processing rounds in a complete encryption process. AES [2] is a typical encryption algorithm with S-P structure.

In the S-P structure, the decryption process is composed of decryption rounds, each component in a decryption round is the reverse of the corresponding component in an encryption round. For the case of AES, each round of encryption is composed of four parts: SubBytes, ShiftRows, MixColumns, and AddRoundKey. Apart from AddRoundKey of which the inverse is itself, other components need to formulate their inverses respectively.

The encryption and decryption processes of AES with 128-bit key can be depicted in Fig. 4.4.

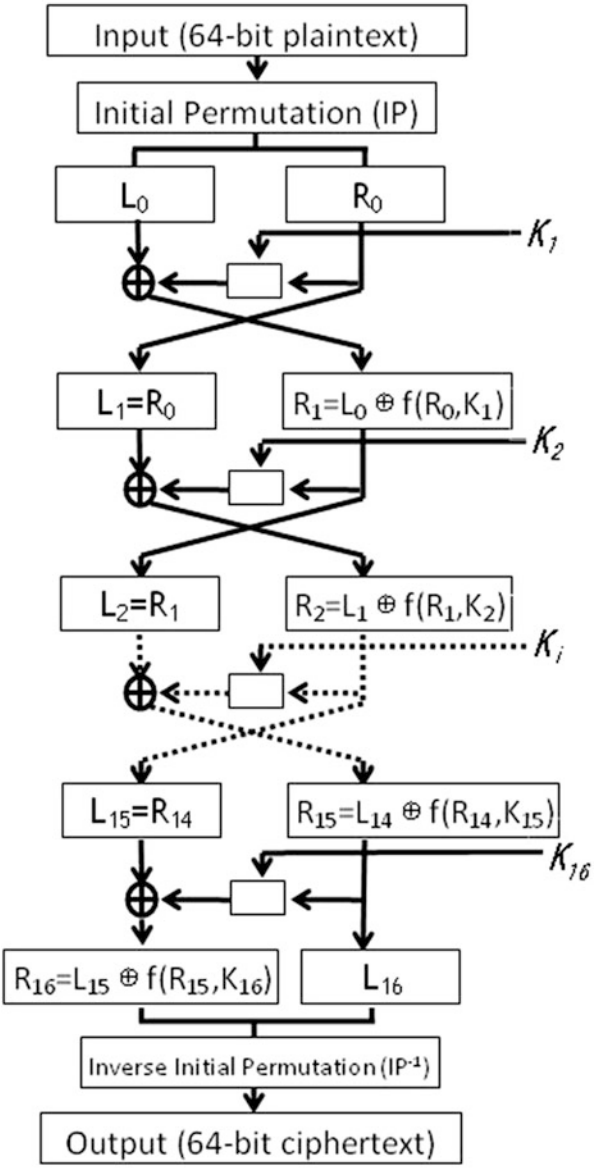

Fig. 4.3 The encryption process of DES

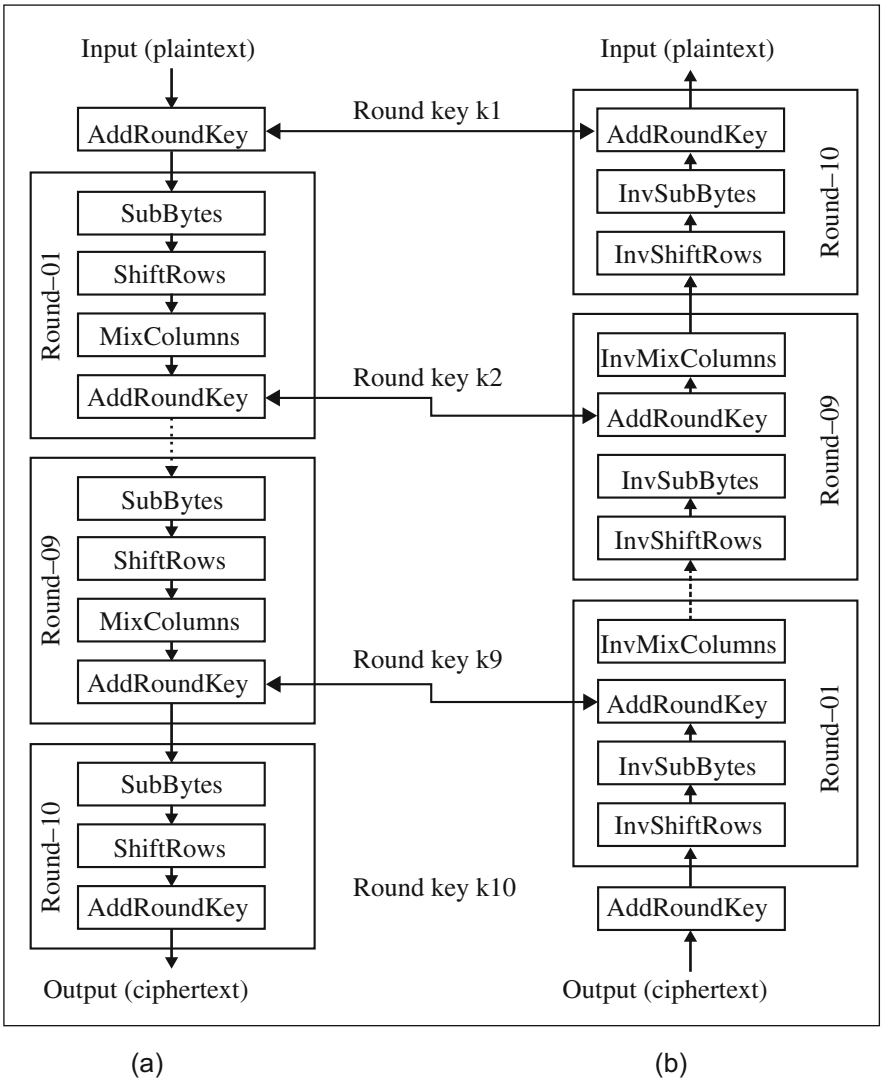

Fig. 4.4 The encryption and decryption processes of AES. (**a**) AES encryption. (**b**) AES decryption

4.2.4 Modes of Operation of Symmetric Block Ciphers

Given a symmetric block cipher, assume that the encryption algorithm is E and the corresponding decryption is D. From Fig. 4.1 it is known that, the plaintext message to be encrypted needs to be split into blocks, with each block being of the same size as the block length of the encryption algorithm. Often padding is needed to make the last block of the plaintext to be of the block length. Intuitively, it can be understood that, the use of the encryption algorithm is to encrypt each block of the plaintext message, and the decryption process is to decrypt each of the ciphertext blocks. This is indeed a way of using a symmetric block cipher in the mode of *electronic codebook (ECB)*, as depicted in Fig. 4.5.

Apart from the ECB mode, there are other modes of operations. The most popular modes of operations are *cipher block chaining (CBC)* mode and *counter (CTR)* mode, they are shown in Figs. 4.6 and 4.7 respectively.

It is noted that the CBC mode makes the encryption of different blocks of plaintext message in a sequential order, while ECB mode can actually use parallel processing on different message blocks. The CTR mode is even special, both the encryption and decryption only use the original encryption algorithm, which means that the encryption algorithms do not have to be invertible, and the effect of encryption and decryption is like that in a stream cipher.

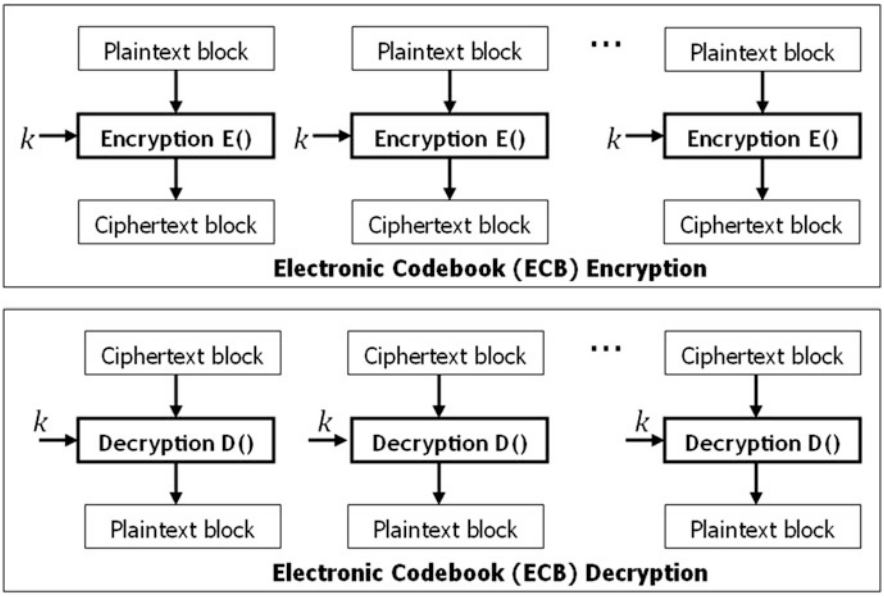

Fig. 4.5 The electronic codebook mode of operation

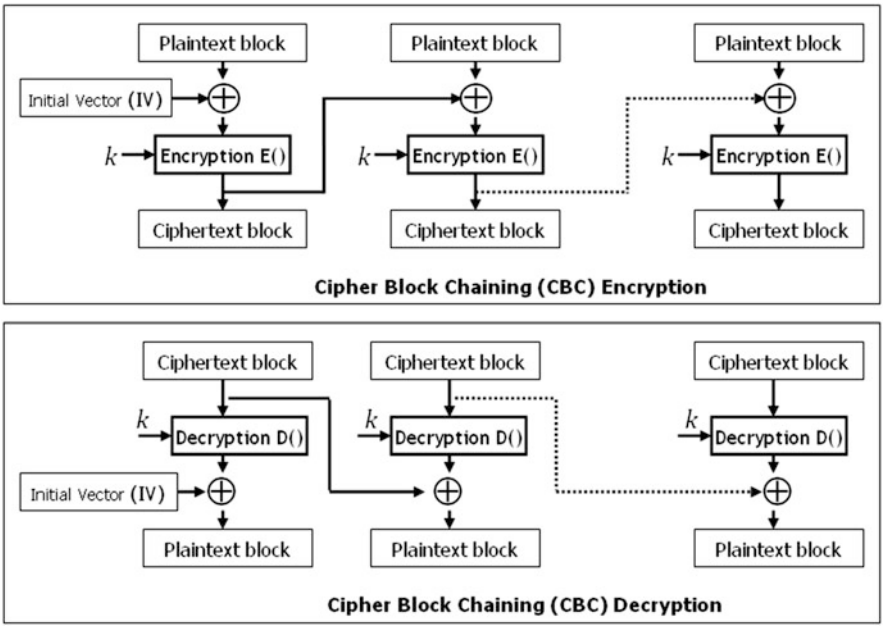

Fig. 4.6 The cipher block chaining (CBC) mode of operation

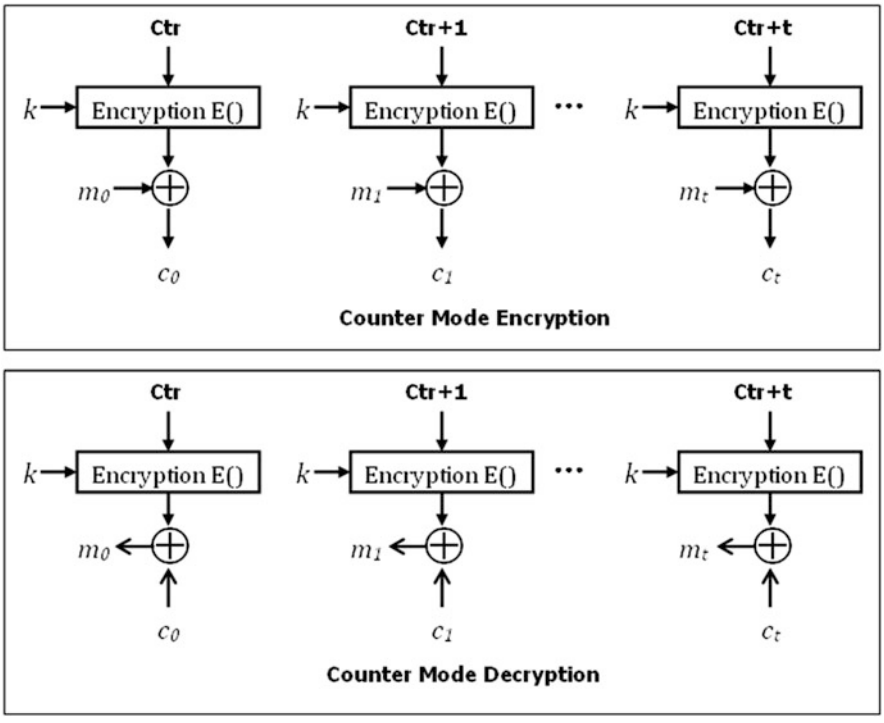

Fig. 4.7 The counter (CTR) mode of operation

4.2.5 Public Key Cryptosystems

In 1976, Diffie and Hellman [10] published a paper titled "New directions in cryptography". The paper indicates that a secret message can be securely shared between two parties by exchanging messages over public networks. Although Diffie and Hellman did not give a concrete construction of public key encryption algorithm, it was then realized that public key encryptions can be constructed, this leads to the publication of many public key cryptosystems.

Different from symmetric key ciphers, an asymmetric key cipher uses different keys in the encryption algorithm and the decryption algorithm. Moreover, in the design of an asymmetric key cipher, it is required that the decryption key cannot be practically computed when the encryption key is given. Given this property, the encryption key can be made public and the encryption algorithm can be used by anyone in the public who wants to encrypt a message, and the ciphertext can only be decrypted with the correct decryption key and the decryption algorithm. Therefore, asymmetric key ciphers are often called public key cipher, or public key cryptosystem, where the later means the whole process of the system design, including how to set the system parameters and the keys, what are encryption and decryption algorithms. The encryption key is hence called the *public key* and the decryption key is called the *private key*.

Denote pk as the public key of a public key cryptosystem, sk is the corresponding private key, E is the encryption algorithm, and D is the decryption algorithm. Then the encryption and decryption process of a public key cipher can be depicted in Fig. 4.8.

Formally, a public key cryptosystem is composed of the following components:

(1) The plaintext space M, which can be represented by binary strings of length n. The plaintext space M may cover all the vectors over $\{0, 1\}^n$.
(2) The ciphertext space C, which can be represented by binary strings of certain length l. In a public key cryptosystem, often l is larger than n, hence the ciphertext space C is a subspace of $\{0, 1\}^l$.
(3) The key space K, which is the set of all possible public key and private key pairs (pk, sk).
(4) The encryption algorithm E, which is a mapping from the plaintext space M to the ciphertext space C.

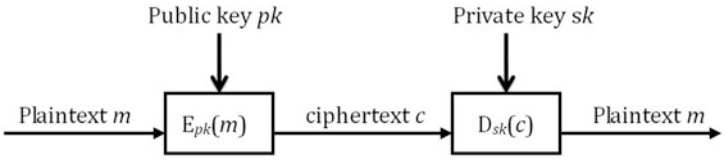

Fig. 4.8 The encryption and decryption process of a public key cipher

(5) The decryption algorithm D, which is a mapping from the ciphertext space C to the plaintext space M.

Many public key cryptosystems have been designed in public literatures. However, only a few are commercially used. One of typical public key cryptosystems is the RSA cryptosystem, which was invented by Rivest, Shamir and Adleman.

4.2.6 The RSA Public Key Cryptosystem

In 1978, Rivest, Shamir and Adleman [11] published a paper titled "A method for obtaining digital signatures and public-key cryptosystems", and later on people refer the work as RSA public key cryptosystem. The paper presents a public key cryptosystem that can be used as public key encryption, and can be used for digital signatures as well. The security of RSA public key cryptosystem was based on integer factorization problem, which is known as a non-deterministic polynomial (NP) problem.

In order to understand how RSA public key cryptosystem works, here gives a brief description about the system.

System setup: In order to set up an RSA public key cryptosystem, one needs to choose two different large prime numbers p and q. Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$.

Key generation: Randomly choose an integer e smaller than and co-prime with $\phi(n)$, compute d such that

$$ed \mod (\phi(n)) \equiv 1. \tag{4.3}$$

Publish (e, n) as the public key, d is kept by the owner user as the private key, and the parameters p and q are useful only in the setup of the system. Once the public key cryptosystem is set up, p and q can be securely discarded.

Encryption and decryption: Given a message m that is smaller than n, the encryption algorithm is

$$c = m^e \mod n, \tag{4.4}$$

and the corresponding decryption algorithm is

$$m = c^d \mod n. \tag{4.5}$$

From the above it can be seen that, the public key e and the modulus n is publicly known. If one can find the inverse of e in the sense of modulo $\phi(n)$, then one can compute the private key d (or a different d', if the inverse is not unique, and d' works the same as d) and hence break the system. However, this problem is related to the factorization of n, and the integer factorization problem is known to be a hard

problem. Apart from factorizing the modulus n, there is no other way known so far to break the RSA system. So the security of RSA cryptosystem is based on the integer factorization problem.

Another important application of RSA public key cryptosystem is to create digital signatures. With the same setup as described above, the digital signature system can be briefly described below:

Digital signature creation: Given a message m that is smaller than n, the public key owner uses his private key d to compute

$$s = m^d \bmod n, \tag{4.6}$$

and s is called the *signature of message* m, which is sent to the verifier (or to be made public, if the verification is done by public) together with the message m.

Signature verification: When someone wants to verify whether s is the digital signature of message m created by the user with public key e, one verifies whether the following equality holds:

$$m = s^e \bmod n? \tag{4.7}$$

It is noted that the signing process uses the private key d, which is similar to the decryption algorithm, but takes the plaintext message m as an input. The signature verification process uses the public key pk, which is similar to the encryption, but takes the signature s as an input, and message m is used for comparison. In this sense, digital signatures do not provide message confidentiality service, because the original message is used for the signature verification.

4.2.7 The Advantages and Disadvantages of Different Types of Ciphers

In the above, different types of ciphers are described. What are the advantages and disadvantages of those ciphers? If one type is better than another, then the applications of the less advanced one would be limited. However, in contemporary secure communications, all of those ciphers are used in different applications, because each type of the ciphers has its own advantages and disadvantages. Some of the basic advantages and disadvantages of different types of ciphers are listed below, which should not be treated as absolute properties, but only a general reference. In fact, it is more appropriate to describe some of the properties as features, because they can be advantages in one scenario, and may be disadvantages in another.

Stream cipher: Stream ciphers are generally quite fast, particularly when being used to encrypt large amount of data. Because most stream ciphers encrypt messages byte-by-byte, they do not cause error propagation, i.e., any error on the ciphertext only affects the correct decryption of the corresponding bytes

of the plaintext. This feature is particularly suitable for wireless transmission. Therefore, wireless communications usually use stream ciphers to encrypt messages. Even if the core encryption algorithm is a block cipher, suitable mode of operation can be used so that the effect of the encryption process is still exclusive-or of plaintext and key stream, with the same effect as a stream cipher.

An obvious disadvantage of stream ciphers is the overhead pre-computation of initialization. When a key and sometimes an initial vector are loaded into the encryption algorithm, the algorithm runs initialization process before producing pseudo-random key stream for data encryption. If the message to be encrypted is small, then the efficiency is greatly reduced, because every time the encryption algorithm is executed, the initialization process goes first, which makes a large proportion of overhead computation.

Another disadvantage of stream ciphers is that most of the existing stream ciphers have severe security flaws. This is perhaps one of the main reasons why the eSTREAM program terminated without a winning candidate to be the standard, and 3GPP (Third Generation Partnership Project) uses block cipher KASUMI to form a stream cipher like function called $f8$, which generates pseudo-random key streams for data encryption.

Symmetric block cipher: Symmetric block ciphers are a class of successful ciphers, many of them have been commercially used. Symmetric block ciphers are quite efficient, with high security, even the outdated DES has no severe security flaws revealed. With the lightweight design, block ciphers can be designed to be more efficient and require small amount of resources (hardware resource and/or software resource) to implement and to execute.

Block ciphers are usually implemented in a specific mode, the CBC mode is one of the recommended mode to use. Although the CBC mode has high error propagation property, this property is deemed to be desirable in many applications.

The disadvantages of symmetric block ciphers are mainly that, communication parties need to setup a common key to enable correct decryption of ciphertexts, i.e., the key agreement problem. This disadvantage can be minimized by the combination of using a public key encryption.

Public key cryptosystem: Public key algorithms also process messages in blocks, hence they are also block ciphers. However, different from symmetric key ciphers, public key ciphers need to setup system parameters individually, hence they are conventionally called *public key cryptosystems*.

The advantage of public key cryptosystems is their convenience. There is no need to agree on a common key for parties to set up a secure communication.

However, the disadvantages of public key cryptosystems are that, public key cryptosystem usually have large computational cost, much slower than symmetric key ciphers. The size of keys (public key and private key) is usually larger than that of traditional symmetric key ciphers. The block size of ciphertext is usually much larger than that of traditional symmetric key ciphers. In general, compared with symmetric key ciphers, public key encryption and decryption algorithms are far less efficient.

4.2.8 Cryptographic Functions and Their Security Services

It is easy to see that, encryption algorithms can provide message confidentiality services, preventing a message to be illegally eavesdropped during its transmission, or illegally accessed during its storage. It is not possible to stop the eavesdropping and illegal access activities, but can prevent the activities to be able to get the content of the protected messages.

The message authentication code algorithms can provide data integrity service. It is not to stop the activity of illegal modification of data, but to enable the illegal modification to be detected.

Digital signature algorithms can be used for data source authentication, i.e., successful signature verification can confirm that the data was created by someone. Apart from that, digital signature algorithms can also provide data integrity service, because any illegal modification of either the original message or the signature will likely to result in the signature verification failure. Another security service that digital signatures can provide is the non-repudiation service, which is a special and quite unique security service.

In information systems, there are many more security requirements and corresponding techniques. IoT devices may adopt some of the security services, and the selection of the security services and implementation techniques are subject to the application requirements and the available resources of the IoT devices.

4.3 Deterministic and Probabilistic Encryptions

Let $E(k, m)$ be an encryption algorithm that takes an encryption key k and a message m as input, the output $c = E(k, m)$ is the corresponding ciphertext. When the encryption key k is fixed, if the message m is always encrypted into the same ciphertext c, regardless how many times the message m is taken as the input to the encryption algorithm $c = E(k, m)$, then the encryption algorithm is called a *deterministic encryption algorithm*. More precisely, a deterministic encryption algorithm will always yield the same output for a same input. It is easy to verify that the encryption algorithms above are all deterministic encryption algorithms, including stream ciphers, symmetric key block ciphers, and the RSA public key encryption algorithm.

However, in practical applications, there is a need for the ciphertext to change dynamically, i.e., the same message m may get different ciphertexts when it is encrypted at different time. Although different initial vectors in some modes of operation of block ciphers can get different ciphertexts for a same message, this change is not dynamic enough. In fact, an initial vector is treated as a supplementary input of the encryption algorithm.

Opposed to the deterministic encryption algorithms, there is another type of encryption algorithms, called *probabilistic encryption algorithms*. An probabilistic encryption algorithm generates an unpredictable output given a same input. This means that, for a probabilistic encryption algorithm E and an encryption key k, given a message m, the ciphertext $c = E(k, m)$ changes unpredictably when the algorithm is executed in different time.

It is noted that the above introduced algorithms are all deterministic encryption algorithms. Are there probabilistic encryption algorithms? the answer is yes.

4.3.1 Elgamal Public Key Cryptosystem

In 1985, Elgamal proposed a public key cryptosystem [12]. The proposed public key cryptosystem is a probabilistic encryption algorithm. The Elgamal public key cryptosystem is based on discrete logarithm problem. The cryptosystem can be described as follows:

Setup: Choose a large prime p and an integer $g < p$. Choose a random number $x < p$ as the user private key, and compute $y = g^x \bmod p$ as the user public key;

Encryption: Given a message $m < p$ and the public key y, the encryption process is as follows:

(1) Randomly choose a number $k < p - 1$ that is co-prime to $p - 1$.
(2) Compute $a = g^k \bmod p$ and $b = y^k m \bmod p$;
(3) The ciphertext is the pair (a, b).

Decryption: Given the ciphertext (a, b) and the private key x, compute $m = b/a^x \bmod p$.

The correctness and security have been throughly studied, and the Elgamal public key cryptosystem can also be used for digital signatures. Here we are only interested in the probabilistic behavior of the encryption algorithm. Note that part of the ciphertext is generated based on a random number k, i.e., $a = g^k \bmod p$. This means that, every time the encryption algorithm is executed on a same message m, a is likely to be a different value, as a consequence, b must also be different from the previous value. It is further noted that, different values of a are statistically independent of each other, if the choices of k are random. This property indicates that Elgamal public key encryption algorithm is indeed a probabilistic encryption algorithm.

4.3.2 Turning a Deterministic Encryption Algorithm into Probabilistic Behavior

While probabilistic public key encryption algorithms can be designed, probabilistic symmetric key encryption algorithms can hardly be designed, because symmetric key encryption algorithms usually consider hardware implementation, and random logic has not been properly used in the design of symmetric key algorithms.

Fortunately, it is possible to make a deterministic encryption algorithm to have probabilistic behavior. It is not to change the encryption algorithm itself, but to change the way of using the algorithm.

Let E be a symmetric key encryption algorithm and is deterministic. Let the block size of the algorithm be n. Given a message m, in order to get the probabilistic behavior of the ciphertexts, the message is split into smaller sizes, e.g., $k(k < n)$ bits. then the encryption and decryption processes can be done as follows:

Encryption: Given a k-bit message m $(k < n)$, the encryption process is as follows:

(1) Generate a random number r in $n - k$ bits;
(2) Compute $c = E(k, m\|r)$, where $m\|r$ is the string concatenation of m and r;
(3) Produce c as the ciphertext of m.

Decryption: Given a ciphertext c, and denote the corresponding decryption algorithm as D, then the decryption process is as follows:

(1) Decrypt c using the decryption algorithm to get $m' = D(k, c)$;
(2) Let the most significant k bits of m' be m, and produce m as the decrypted message.

It is obvious that the ciphertext generated by the above encryption process is probabilistic, i.e., a same plaintext message m will get different ciphertexts when it is encrypted in different time, this is due to the involvement of the random number r. However, the randomness of the ciphertexts are limited, i.e., a same plaintext may have up to 2^{n-k} possible ciphertexts under the same encryption key. If $n - k$ is large, then k must be relatively small, which means that the encryption efficiency is reduced, and the overhead processing further decreases the efficiency of the algorithm, this is the sacrifice of the algorithm for the randomness behavior.

The above process of using a deterministic encryption algorithm is suitable only for messages of small size, particularly when the size of messages is smaller than the block size. There are practical applications where the messages are of small size. For example, the identities of IoT devices can be represented by 64 bits or less, and the block size of most block ciphers is 128 bits, this allows the message encryption to attach a 64-bit random number, making the ciphertexts sufficiently random.

4.4 Cryptographic Hash Functions and Message Authentication Codes

4.4.1 Cryptographic Hash Functions

There is an important class of cryptographic functions called *hash functions*. A cryptographic hash function (denoted by $H(x)$) should have the following properties:

Compression. A hash function should have fixed length of outputs for arbitrary length inputs. Given the fact that, in most applications, the length of input message is far larger than that of the output, the hash function acts like a compression function. However, in some extreme cases when the length of the input message is less than that of the output, the effect of the hash function is not a compression, but an expansion.

One-way. Let $H(x)$ be a cryptographic hash function. Given any x, it should be easy to compute the result $H(x)$. However, given any value $y = H(x)$, it is computationally infeasible to find an input x satisfying that $H(x) = y$ holds. This property is called *pre-image resistance*. Note that the pre-image is not unique, so the computation of a pre-image of $y = H(x)$ is not necessarily the original message x.

Weak collision resistance. Given an input x and its hash value $H(x)$, it should be computationally infeasible to find $x' \neq x$ such that the equality $H(x') = H(x)$ holds. This property is called *second pre-image resistance*.

Strong collision resistance. If it is computationally infeasible to find two different values $x \neq y$ such that $H(x) = H(y)$ holds, then the hash function $H()$ is said to have *strong collision resistance*. This property is closely related to the problem of "birthday attack", which says that the computational cost to find a collision $x \neq y$ such that $H(x) = H(y)$ holds, is about $2^{length(H)/2}$, i.e., 2 to the power of half the length of the hash output. This gives a good reference to set an appropriate length of the output of a hash function.

The first successful commercially used hash function is MD5 [13]. The output of MD5 is 128 bits, hence the computational cost to find a strong collision is $2^{128/2} = 2^{64}$ which is quite a large number. However, sophisticated analysis often can find better methods to find collisions, and MD5 is now treated as insecure due to collision can be found with affordable computation.

Other typical hash functions include a number of series of secure hash algorithm standards made by NIST, namely SHA-1, SHA-2, and SHA-3. SHA1 produces 160 bits of output, and some weakness of SHA-1 has been found. SHA-2 has different versions with different lengths of output, the most popular versions are SHA-256 and SHA-512, and other less popular versions include SHA-224 and SHA-384. SHA-3 is a new hash function standard which is meant to be more efficient than SHA-2.

4.4.2 Applications of Hash Functions: (1) Message Authentication Codes

A hash function works like a compression function. Together with its other cryptographic properties, hash functions are an important class of cryptographic functions. One of the applications of hash functions is to construct message authentication code (MAC) algorithms.

Message authentication code (MAC) is such a function that maps a message space M into a larger space, denote it as Ω, where the mapping is controlled by a key as depicted in Fig. 4.9.

Practically, this MAC space is created by attaching a specially generated tag to the original message. This special tag is called message authentication code, which is generated by a MAC algorithm taking the original message and a key as input.

There are different constructions of MAC algorithms, however, the most efficient construction is based on hash functions. Let m be a plaintext message, $H(x)$ be a hash function that produces hash code with length r. Then we can define a mapping $x \rightarrow x||H(x,k)$ from the message space $\{0,1\}^n$ into $\Omega = \{0,1\}^{n+r}$, where the operation "$\|$" is string concatenation. If one knows the key k, then it is easy to verify whether a message authentication code $x||h$ is the correct MAC of message x. The verification process is simple: take x and the key k as input, compute the hash output h', compare whether the computed h' and the h from the received message $x||h$ are the same. If so, then the message is accepted, otherwise, the message is suspected as having been illegally modified, and should be rejected. It is easy to see that, any illegal modification on x or h will likely to result in that

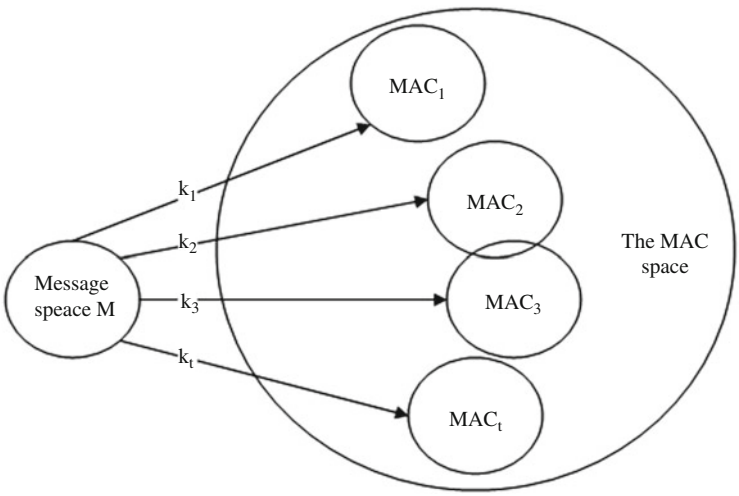

Fig. 4.9 The message authentication code is a mapping from original message space into a larger message space

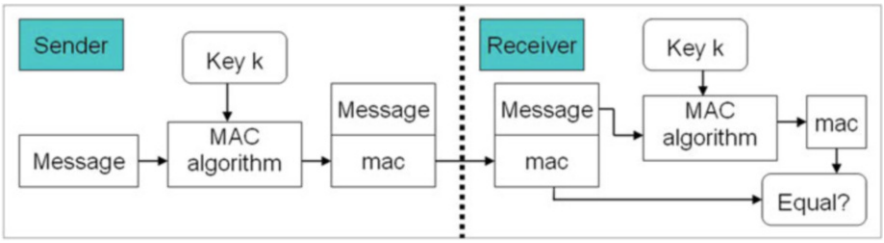

Fig. 4.10 The process of using message authentication code

$H(x', k) \neq h'$, where x' is the message with illegal modification, and h' is the MAC with illegal modification. Without knowing the key k, the chances of successful illegal modification are negligible, hence can be detected. The process of using message authentication code to provide message integrity service can be depicted in Fig. 4.10.

The above designed MAC function $H(x, k)$ is just used to explain how an MAC algorithm works. In modern cryptography, a more sophisticated design of MAC algorithm is called HMAC [14], which has been proved to be sufficiently secure. HMAC is based on an existing hash function $H(x)$. Given a message m and a key k, the HMAC is defined as

$$HMAC(k, m) = H(k \oplus opad \| H(k \oplus ipad \| m)) \tag{4.8}$$

where *opad* and *ipad* are two constants defined by replicates of 0x5c and 0x36, $\oplus$ is the logical operation exclusive-or, $\|$ is the concatenation operation, and the length of the key k should be 64 bytes. If the length of k is less than 64 bytes, then simply padding k with as many 0's as needed.

4.4.3 *Applications of Hash Functions: (2) Digital Signature Algorithms*

Digital signature algorithms are a class of important cryptographic algorithms. Digital signature algorithms are usually based on public cryptosystem, hence a digital signature algorithm can be treated as an application of public key cryptosystem.

When a digital signature is created for a given message m, the signer uses his private key and employ the signing algorithm to produce the signature s. When a signature s is verified, the original message m and the signer's public key should be given. A verification algorithm is used to verify whether the signature s is a valid signature on m created by the signer. The process of signature creation and verification can be depicted in Fig. 4.11.

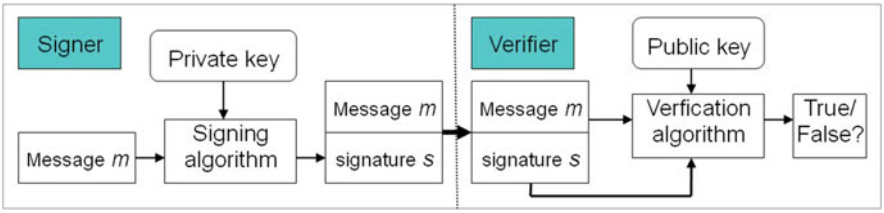

Fig. 4.11 The creation and verification of a digital signature

Similar to manual signatures, a digital signature should satisfy the following properties:

Verifiability: A digital signature is verifiable by whoever having the public key information.

Unforgeability: Without the signer's private key, no one is able to forge a signature s' on any message m' that can pass the signature verification process.

Although a digital signature algorithm can be treated as an application of a public key cryptosystem, however, not every public key cryptosystem can be used for digital signatures. The McEliece public key cryptosystem [15] is inappropriate to be used for digital signatures.

It is known from Sect. 4.2.6 that, for an RSA cryptosystem with user public key (e, n) and private key d, for a message m, the signature creation algorithm is $s = m^d \bmod n$. Given the signature s and the original message m, the verification process is to check whether the equality $s^e \bmod n = m$ holds.

Note that the above signature algorithm works only when $m < n$ holds. If the message to be signed is larger than n, then the message has to be split into parts with each part being smaller than n. For a large message (e.g. a digital video with high resolution), the signing process is inefficient. For example, using an RSA public key to encrypt a 2-gigabyte video, assuming that the RSA modulus is 1024 bits (128 bytes), then the video file has to be split into at least $2^{23} = 8{,}388{,}608$ parts, each part is signed separately. It is obvious that the signing process is slow, and the size of the signature is at least the same as the original video file. The same problem exists for the signature verification process. Moreover, there is a possibility that the signature can be claimed as a valid signature of a re-join of the message parts if it represents a different meaning, which violates the message integrity, and is deviated from the initial purpose of digital signatures.

Fortunately, a hash function can be used to overcome all these drawbacks. Let $H(x)$ be a cryptographic hash function. Given message m, the signing algorithm of Eq. (4.6) can be modified as follows:

(1) Compute the hash code of m, i.e., $h = H(m)$;
(2) Compute the RSA signature on h:

$$s = H(m)^d \bmod n. \tag{4.9}$$

(3) The signature of m is $m \| s$.

Given an RSA signature $m \| s$ generated above, the corresponding signature verification process is as follows:

(1) Extract m from the signature and compute its hash code $h = H(m)$;
(2) Extract s from the signature and check whether the following equality holds:

$$s^e \bmod n = H(m)? \tag{4.10}$$

It is easy to verify that, by employing a hash function in digital signature algorithm, the message integrity is ensured, the efficiency of signature creation and verification is greatly improved, and the size of the signature does not depend on the size of the original message.

It should be pointed out that, apart from RSA digital signatures, there are many other digital signature algorithms. The digital signature standard (DSS) [16] is a standard for digital signatures in commercial applications.

4.5 The Lightweight Features of Cryptographic Algorithms

Traditional cryptographic algorithms are designed for achieving high-level security. Other properties such as efficiency and implementation cost are considered as attracting properties but not a strict measure in the algorithm design. With the development of IoT applications, and the fact that large amount of IoT devices are resource-constrained, it is desirable to design lightweight security techniques [17, 18].

However, for resource-constrained devices such as RFID tags and some low cost sensors, it is not easy to implement sufficient cryptographic functions. In order to provide security services for resource-constrained devices, some cryptographic algorithms (e.g., [19–21]) are designed with low implementation cost, such algorithms are called *lightweight* cryptographic algorithms.

There is a good description about the properties of lightweight cryptography in ISO/IEC 29192 in ISO/IEC JTC 1/SC 27. However, the term "lightweight" is not a technical measure, but a descriptive term.

The development and security requirements of IoT application systems and the resource-constraint feature of many IoT devices put forward the demand of lightweight cryptography. While a number of lightweight encryption algorithms have been designed, the study on lightweight hash functions and message authentication codes, lightweight public key cryptography, particularly lightweight digital signatures, progressed slowly. Due to the similarity between the design of hash functions and that of encryption algorithms, the design of lightweight hash functions and hence that of lightweight massage authentication codes is not a big challenge. However, the design of lightweight public key cryptography remains a big challenge, because public key cryptosystems are usually based on an open hard mathematical problem, while the size of parameters (e.g., a modulus, prime

numbers) of the problem cannot be significantly small because they directly affect the security of the cryptosystem. It is proposed that, the design of imbalanced public key cryptosystem might be possible. By saying *imbalanced*, it means that either the encryption algorithm consumes little computation resources while the corresponding decryption algorithm requires large amount of computation, or the other way round. Similar principle applies to the design of imbalanced digital signatures. This trade-off would maintain a good security level, and squeezes some computing savings for one party by sacrificing the other party's resources.

Compared with the lightweight cryptographic algorithms, the design of lightweight cryptographic protocols is more influential to practical IoT applications, because cryptographic protocols are related to computation and communication resources, where the energy consumption of communication is usually much more than that of cryptographic computation on the same amount of data.

4.6 Summary

Cryptographic techniques are fundamentals to data security, both in data storage and data transmission. Some basic concepts and mechanisms of cryptographic techniques are introduced in this chapter, including the encryption algorithms, message authentication code algorithms, digital signature algorithms, and basic methods of identity authentication protocols. Different cryptographic techniques are designed to provide different security services. The encryption algorithms provide data confidentiality service, which means that unauthorized users cannot get the content of the message even if they can get the message; The message authentication codes provide data integrity service, which means that illegal modification on the data can be detected with overwhelming probability; Digital signature algorithms provide non-repudiation service, which means that the digital signature is evidence about someone's consent on the content of some data.

However, as can be seen from the description and analysis in this chapter, a same cryptographic function may provide multiple security services, while a specific security service can be provided by different cryptographic methods.

It is noted that, the description of the cryptographic techniques in this chapter is mostly conceptual, more sophisticated description is beyond the scope of this book, because the focus of this book is about the security services provided by cryptographic methods, instead of techniques in cryptography.

This chapter also gives a brief description about some security protocols, including key agreement protocols and authentication protocols. Apart from such security protocols, there are some commercial standard security protocols, the most commonly used network security protocols are IPSec, SSL/TLS, SET, PGP, etc.. Many network applications have integrated IPSec or SSL/TLS to form security-enabled applications. These network security protocols are related to network layer security of IoT.

References

1. Data Encryption Standard, FIPS PUB 46, National Technical Information Service, VA (1977)
2. Advanced Encryption Standard (AES), NIST FIPS 197 (2001). https://doi.org/10.6028/NIST.FIPS.197. Visited on 10 Sep 2020
3. International data encryption algorithm. https://www.includehelp.com/cryptography/international-data-encryption-algorithm-idea.aspx. Visited on 10 Sep. 2020
4. RC2 symmetric block cipher. http://www.crypto-it.net/eng/symmetric/rc2.html Visited 10 Sep. 2020
5. The SM4 block cipher algorithm and its modes of operations, IETF draft (2018). https://tools.ietf.org/id/draft-crypto-sm4-00.html. Visited on 10 Sep. 2020
6. RC4 encryption algorithm. https://simple.wikipedia.org/wiki/RC4. Visited on 10 Sep. 2020
7. A5 stream cipher. https://asecuritysite.com/encryption/a5. Visited on 10 Sep 2020
8. The stream cipher ZUC. http://www.ccsa.org.cn/english/zuc.htm. Visited on 10 Sep 2020
9. The eSTREAM Project. https://www.ecrypt.eu.org/stream/allcandidates.html. Visited on 10 Sep. 2020
10. W. Diffie, M.E. Hellman, New Directions in cryptography. IEEE Trans. Inf. Theory **IT-22**, 644–654 (1976)
11. R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
12. T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **IT-31**(4), 469–472 (1985)
13. R.L. Rivest, The MD5 message-digest algorithm, Internet Request for Comments. Internet Request for Comments (RFC) 1321 (1992)
14. H. Krawczyk, M. Bellare, R. Canetti, *HMAC: Keyed-Hashing for Message Authentication.* IETF RFC2104. https://www.rfc-editor.org/info/rfc2104. https://doi.org/10.17487/RFC2104
15. R.L. McEliece, A public-key cryptosystem based on algebraic coding theory. Deep Space Network Progress Report 42-44, Jet Propulsion Labs, Pasadena (1978), pp. 114–116
16. National Institute for Standards and Technology, Digital Signature Standard (DSS). Federal Reg. **56**(169) (1991)
17. S. Oza, D. Mathpal. A study on Internet of things security and lightweight cryptography. Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol. **3**(3), 683–689 (2018)
18. G. Yang, B. Zhu, V. Suder, M.D. Aagaard, G. Gong, The Simeck family of lightweight block ciphers, in *Cryptographic Hardware and Embedded Systems (CHES 2015)*. Lecture Notes in Computer Science, vol. 9293 (Springer, Berlin, 2015), pp. 307–329
19. A. Bogdanov, L.R. Knudsen, G. Leander, et al., PRESENT: An ultra-lightweight block cipher, in *Proceedings of 9th International Workshop on Cryptographic Hardware and Embedded Systems - CHESS 2007*. Lecture Notes in Computer Science, vol. 4727 (Springer, Berlin, 2007), pp. 450–466
20. W. Wu, L. Zhang, LBlock: A lightweight block cipher, in *Proceedings of 9th International Conference on Applied Cryptography and Network Security*. Lecture Notes in Computer Science, vol. 6715 (Springer, Berlin, 2011), pp. 327–344
21. B. Rashidi, Flexible structures of lightweight block ciphers PRESENT, SIMON and LED. IET Circuits Devices Syst. **14**(3), 369–380 (2020).

Chapter 5
Trust Mechanism and Key Management in IoT

In the IoT security architecture described in Chap. 3, there is a supporting pillar called trust and key management. This chapter discusses what "trust" is in the sense of network environment, and how such "trust" can be established and extended/transferred. Based on the "trust", secret keys can be shared securely and reliably, i.e., some attacks based on identity forgery can be identified, such as man-in-the-middle attacks. Different types of identities are introduced (e.g. account as an ID, address as an ID, tag as an ID), and some mechanisms of identity authentication are introduced. The "lightweight" feature of identity authentication is addressed. Such feature is to accommodate the security requirements of resource-constrained IoT devices.

5.1 Introduction

In a secure communication system using a symmetric encryption, both the sender and the receiver have to share a secret key, so that the sender can encrypt plaintext messages and the receiver can decrypt the ciphertext messages. Such a secret key can be sent via a secure channel, which is costly and does not suit commercial applications. Diffie-Hellman key agreement protocol provides a way to allow a sender and a receiver to agree on a common secret over public networks. Using a public key cryptosystem, secret keys can be sent using public key encryption without the need of message exchange, which is more efficient.

While the symmetric key agreement problem can be solved by using a public key cryptosystem, or by Diffie-Hellman key agreement protocol, how to manage public keys becomes a new problem. Active attacks such as impersonation attack and man-in-the-middle attack are difficult to avoid. However, such attacks can be protected by adding an authentication service, and the authentication service has to first solve such a problem: how to validate the identity of a communication party on

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_5

the other end of the network? i.e., how to confirm someone to be the one it claims to be? There must exist some kind of trust, otherwise, reliable authentication cannot be done.

What is trust? The Oxford dictionary explains "trust" as "firm belief in the reliability, truth, or ability of someone or something". This explanation also applies to the network environment. However, it should be noted that, the meaning of "trust" must be associated with certain circumstance and context. For example, the relation of trust for a patient on a doctor is different from that for a bank account. The meaning of "trust" in a network environment is that one communication party has something that can be used to confirm whether the other party is really the one it claims to be. In this scenario, the one who proves to be someone (the claimed one) is called *the prover*, and the one who verifies whether the claimed identity to be genuine is called *the verifier*. The verifier can judge whether the prover is the one with the claimed identity, if and only if the verifier *trusts* on the prover, i.e., the verifier has something (e.g., some knowledge) that only the genuine prover can prove the knowledge without actually presenting the knowledge.

There has been some study on the trust mechanism in different applications [1, 4, 5, 8]. However, most of the study on the "trust" relation is from the social aspect. The "trust" relation in this book focuses on the technical aspect, i.e., the relation is about holding a piece of special information (e.g., the knowledge) that can be used to verify the authenticity of another party, while the trust on the authenticity of that piece of information is from social aspect.

To make the concept of the relation of trust more explicit, we tentatively give the following definition.

Definition 5.1 Let A and B be two entities connected via communication networks. We say that A trusts on B, if A has some authentic information (a special knowledge) that can be used to verify that someone knows the secret information which is supposed to be known by B exclusively (except A).

The above definition has two key parts: (1) A has some *authentic information*, which cannot be a forgery; (2) That authentic information can be used to verify whether the other party is the holder of B's secret. The definition does not say how the verification can be done, this is what to be solved by authentication protocols.

5.2 Properties of the Relation of Trust

From Definition 5.1 it can be seen that, the relation of trust is a one-way relation, and does not automatically mean the reverse. This is to say that, A trusts on B does not mean that B also trusts on A. For convenience of writing, the relation that A trusts on B is denoted as $A \xrightarrow{trust} B$.

It is easy to verify that the relation of trust has the following properties:

(1) **One-way:** The relation of trust is one-way, i.e., $A \xrightarrow{trust} B$ does not mean that $B \xrightarrow{trust} A$ also holds.
(2) **Transferability:** The relation of trust can be transferred. If the relations $A \xrightarrow{trust} B$ and $B \xrightarrow{trust} C$ hold, then there exists some mechanism to establish the relation $A \xrightarrow{trust} C$.
(3) **Starting point:** In the network environment, trust can be transferred, but cannot be established from nothing. There must be a starting point for the relation of trust to be established, and the starting point is called the *initial relation of trust*, or simply *initial trust*.
(4) **Initial trust:** The establishment of initial trust is usually done off-line. If a trusted third party (TTP) is available, then the relation of trust between A and B can be established. However, this is actually the result of transfer of trust.

Definition 5.2 If there exist intermediate entities $A_1, A_2, \cdots, A_k$ such that the relations $A \xrightarrow{trust} A_1$, $A_1 \xrightarrow{trust} A_2, \cdots, A_{k-1} \xrightarrow{trust} A_k$, $A_k \xrightarrow{trust} B$ hold, then we say that there is a *path of trust* from A to B. As an extreme case, there exists a path of trust from any party to itself, which is called the *degenerated path of trust*.

By Definition 5.2, it is easy to get the following intuitive conclusion:

Theorem 5.1 *The relation $A \xrightarrow{trust} B$ can be established, if and only there exists a path of trust from A to B.*

Proof **Necessity:** If the relation $A \xrightarrow{trust} B$ can be established, this means that the relation $A \xrightarrow{trust} B$ exists. By Definition 5.2, there exists a path of trust from A to B.

Sufficiency: If there exists a path of trust from A to B, by Definition 5.2, there are some intermediate entities $A_1, A_2, \cdots, A_k$ such that the relations $A \xrightarrow{trust} A_1$, $A_1 \xrightarrow{trust} A_2, \cdots, A_{k-1} \xrightarrow{trust} A_k$, $A_k \xrightarrow{trust} B$ hold. By the transferability property of the relation of trust, the relations $A \xrightarrow{trust} A_1$ and $A_1 \xrightarrow{trust} A_2$ indicate the existence of the relation $A \xrightarrow{trust} A_2$, together with the relation $A_2 \xrightarrow{trust} A_3$, it means the existence of the relation $A \xrightarrow{trust} A_3$. This relation can be continuously extended till the relation $A \xrightarrow{trust} B$, which proves the sufficiency of Theorem 5.1. □

If there exists a relation of trust $A \xrightarrow{trust} B$, then A can verify whether B is the true one. The process of such a verification is usually done by an authentication protocol. On the contrary, without the relation of trust, there is no base for A to reliably verify whether a communication party is indeed B, assuming that B's secrets for the purpose of identity authentication have not been leaked.

5.3 How to Set Up an Initial Trust

From the above description it can be seen that, new relation of trust can be established based on some existing such relations. Before any new relation of trust can be extended, the existing such relations must be initial relations of trust.

The initial relations of trust can be set up based on symmetric key or public key cryptosystems. Different methods have different mechanism and advantages/disadvantages.

5.3.1 Initial Trust Set-Up Based on Symmetric Key Cryptosystem

With a symmetric key cryptosystem, if A and B share a secret key k which is made in a trusted manner, e.g., the common key is established off-line where forgery is impossible, then the initial relation of trust $A \xrightarrow{trust} B$ is established. Checking with the definition of trust (Definition 5.1), it can be seen that, A has the secret information k which is authentic (since it is set in a trusted manner) and can be used to verify whether someone knows the secret k, which is supposed to be known exclusively by B (except A). Similarly, the initial relation of trust $B \xrightarrow{trust} A$ is also established. Therefore, the off-line secret key sharing establishes a mutual relation of trust between A and B.

An advantage of symmetric key based initial trust establishment is that, mutual trust can be established at the same time. An example of this kind of initial trust establishment is the seed key in a SIM card (the core part of a user mobile device), which was burned into the chip of the SIM during its production. The restriction of symmetric key based initial trust setup is that the setup process is usually made by one party, which suits the applications with a server provider and many users.

The disadvantage is the cost of initial trust establishment, i.e., the management of the shared secret key. Since each IoT device has a unique secret key, the process of writing the secret keys into IoT devices has to maintain a high security level to keep the secret keys confidential.

5.3.2 Initial Trust Set-Up Based on a Public Key Cryptosystem

An initial relation of trust can be established based on a public key cryptosystem. If A has B's public key, and can ensure that it is indeed B's public key, then this means that A has established the relation of trust on B. How can A ensure that a public key belongs to B? This can be done off-line, or by using the service of a public key certificate. For example, A has B's public key certificate and can verify the authenticity of the certificate. Note that the relation of trust established based

on public key cryptography is not mutual, i.e., the initial trust $A \xrightarrow{trust} B$ does not naturally mean that $B \xrightarrow{trust} A$ is also established, this is a disadvantage.

The advantage of initial trust establishment based on public keys cryptosystem is the convenience of using public key certificates. The public key self-certificate of an CA (certification authority) can be written into an IoT device during its manufacturing. Different from writing secret keys, the process of writing a public key certificate does not need to keep its confidentiality, and a same public key certificate can be used for a large number of IoT devices.

Once the relation of trust is established, how to verify the authenticity of the identity of the party on the other end of a network connection is the job of authentication protocols. Once authentication is done, then communication parties can establish shared keys for secure communication, and the process of authentication is usually associated with key establishment (by the way of key agreement or key distribution).

5.4 Types of Identities in IoT Systems

The purpose of authentication is to verify the authenticity of a claimed identity, i.e., to verify that the party who is communicating with the verifier is indeed the one having the claimed identity.

In network environment, every communication party has an identity, regardless whether the party is a machine or a person. Where does the identity come from? Who defined the identity? What are the rules to define an identity? These questions do not have explicit answers. There are different types of entities, and their identities are of different types, and hence should follow different rules when defining different types of identities.

In IoT applications, there can be three types of entities, a name, an address, or a string.

When an identity is represented by a name, it is called a *named identity*. This kind of identities can represent human users, physical devices, and service provision platforms. The difference between a named identity and a name in the real world is that, the identity is unique within the range that the identity could possibly be used, including the case where the identity is globally unique, while a name in the real world may not be unique.

When an identity is represented by an IP address, or a MAC address, then the identity is called a *address-based identity*. The address-based identities also have the property of uniqueness, and in the Internet environment, this kind of addresses should be globally unique.

When an identity is represented by a string, then the identity is called a *tag-based identity*, or simply *tag identity*. There are different kinds of tag identities, typical such identities include bar codes, QR codes, and the codes of radio frequency identification (RFID) tags.

5.4.1 Named Identities

The named identities are the most general type of identities. Apart from some service providers, the most commonly used named identities are accounts for human users.

Within a specific application platform, the user accounts are the named identities. Note that the user accounts should be unique on the specific application platform.

However, many application platforms provide services to anonymous users, i.e., there is no link between the real identity of the user and the user account, and one user may have multiple user accounts on a same application platform.

Moreover, beyond an application platform, there can be many user accounts that share a same account name. Hence, to use the account names as identities is feasible only within a specific application. Considering the IoT applications from global viewpoint, identities should be globally unique, and there should be an efficient way to link a named identity to the real identity of a specific user. One such method is to verify the real identity of a user during the registration phase, and record the linkage between the account name and the real identity of the user. On the other hand, assuming that the identity of any application platform is globally unique, which is usually the case in practice, then the global uniqueness of the named identity can be ensured by the combination of the account name and the identity of the platform. If user accounts are mapped to the real identities of users, then the national unique identity number together with the country code form a global unique identity.

5.4.2 Address-Based Identities

Address-based identities are assigned to the networked devices that have computing and communication capabilities. Computers, routers, communication equipment, and many kinds of IoT devices are of this kind. In order to uniform the address-based identities, IPv6 addresses can be a good solution for those that are connected to the Internet.

The difference between address-based identities and named identities is that, the former can be used to find where the devices are on the network, and the latter are just identities, they have to be associated with a network address in order to communicate.

With respect to the mobile communication devices such as mobile phones, they also have globally unique identities. The international mobile subscriber identity (IMSI) can be used to find where the device is, hence IMSI's are a kind of address-based identities, which have global uniqueness property.

5.4.3 Tag Identities

There are many things in the physical world that do not have communication or even computation capabilities. When such things are managed via the Internet, a tag can be used to identify them. The most popular kind of tags are the RFID tags. Since an RFID tag is more than a digital string representing an identity, it has some short distance communication and limited computation capabilities, hence the most representative tag identities are those represented by RFID tags.

Tag identities look very much similar to name identities. However, they have the following differences:

(1) The identities are provided in different ways. Named identities are usually provided by a human user or a computer on the request of an application, and the process is usually followed by identity authentication. While the tag identities are provided by an RFID reader which has the right to access a database managing the RFID information, and the process is usually not followed by identity authentication.
(2) The authentication mechanisms are different. The mechanism of authentication for named identities is to verify whether the target party has some specific secret information, and the challenge-response mechanism can be used, where the process of verification does not reveal the secret. However, tag identities are just used for finding whether the identity information exists in a specific database. Certain techniques are invented to protect the communication between tags and the readers/writers, and some authentication techniques for RFID tags are invented to avoid tag cloning.

There are a number of coding methods that make the identities of RFID tags globally unique (e.g., EPCglobal), such identities can be properly used in IoT applications regardless of the computing platforms that manage the identities of those RFID tags.

5.4.4 The Authenticity of Identities

Identities are defined for the purpose of proper communication in network environment. However, there are malicious users who may create fake identities or to masquerade existing identities, hence it is important to make sure that the claimed identity is authentic.

Different types of identities have different mechanisms of authentication. The authenticity of named identity means that the messages really come from the party having that name, the authenticity of the address-based identity means that the message really comes from that address, and the authenticity of a tag identity means that there exists some description about the thing that the tag identity is attached to. It should be noted that the way to attach an RFID tag to a physical thing can be hard attachment, e.g., stick them together, or soft connection such as a name tag.

5.4.5 The Authenticity of Anonymous Identities

Although the authenticity of an identity is to verify whether the party owns a specific identity, however, there are cases where the real identities are hidden. In this case, the authenticity of an identity becomes a little more complicated. An anonymous identity authentication scheme should satisfy the following properties:

(1) the authenticity of the identity can be verified, and
(2) the anonymity of the identity is to be maintained.

To maintain the anonymity of an identity while verifying its authenticity is not an easy task. If the purpose of anonymity is just to hid the real identity, then a simple encryption can solve the problem. If the anonymity means the link between an identity and its activities (e.g., the trace of a moving object), then a static encryption does not solve the problem, because the encrypted form of identity can be treated as a new identity representing the same target object. Hence, using probabilistic encryption, or mapping the identity to random strings after each time of successful authentication, are the common methods of anonymous authentication.

As an example in practical applications, the use of temporary mobile subscriber identity (TMSI) is an anonymous authentication technique used in mobile communication networks.

5.5 Identity Authentication Protocols

Once identities are defined, whether an identity is authorized can be verified. However, in network environment, identity impersonation is a common attack. In order to verify whether the communication party really is the one with claimed identity, identity authentication is needed. The most effective method of identity authentication is to use an authentication protocol. There are a large variety of authentication protocols, including simple ones and complicated ones.

5.5.1 Methods of Authentication

The purpose of authentication is to verify and confirm that the remote communication entity indeed represents the claimed identity. By saying that someone represents the claimed identity, it means that the responder can present something that only the claimed identity is supposed to know, to have, or to be.

More specifically, authentication is to confirm the following:

Something you know: One such example is your password, or a secret key that you know. The process of checking something you know is not to let you provide the secret, but to use the secret to generate something that cannot be forged by

others without knowing the secret, and it can be verified by the verifier who may or may not know the secret.

To check what you know, it can be done by using a symmetric key cryptosystem or a public key cryptosystem. If a symmetric key cryptosystem is used, then the verifier needs to share a secret with the user, and the user should prove the knowledge of that secret; If a public key cryptosystem is used, then the verifier is supposed to know and trust the public key of the user. The process of identity authentication is to let the user prove the knowledge of the corresponding private key. The process of identity authentication should not reveal any information about the secret, either a secret key or a private key.

There are different methods to check what you know. The most common method is to use the challenge-response mechanism, which works as follows: the verifier generates a message called *challenge*, and sends the message to the prover. The prover uses the secret and the challenge to compute a message called *response*, and sends the response to the verifier. The verifier checks whether the response is correct. This authentication makes sense only when the following conditions apply:

(1) Only the party having the secret is able to computer a correct response;
(2) The verifier is able to check whether the response is correct;
(3) The challenge changes every time of authentication.

Something you have: One such example is when a swipe card is used to access a secure area, a bank account, or to operate some machine. Information used to identify the swipe card should have been stored in the server where the swipe card is verified. The process of swiping the card is to read the information in the swipe card, compare the information with what is stored in a database, and determine whether access is permitted. The information in the swipe card may or may not have security protection.

To check what you have, it is assumed that the device possessed by the user is able to provide the correct computation result using a secret key embedded into the device, and the verifier (normally a data center) shares the same secret with that device.

The mechanism of checking what you have is to check whether the encryption made by the device is correct. A secure authentication protocol to check what you have usually satisfy the following conditions:

(1) Only the authorized device is able to encrypt designated messages correctly;
(2) The verifier (e.g., a data center) can decrypt the ciphertexts generated by the device;
(3) Some randomness (e.g., a timestamp) should be introduced into the encryption, so that messages do not repeat.

Something you are: This is where biometric security comes in. One such example is your fingerprint. To access a specific data center, your biometric information (e.g., your fingerprints) should be collected and stored in an authentication database. The process of biometric information based authentication is to read

your fingerprint, and compare with what has been stored in the database. If a match is found, then access is permitted. This comparison is fuzzy matching, because precise matching is not practically possible.
To check what you are, it is to verify the biometric information, where the verifier had some biometric information of the user in advance, and the verification process is to make a fuzzy matching, assuming that illegal users are unable to provide the correct biometric information, although this assumption is not always true in practice.

In some IoT applications, more than one authentication mechanism is needed. This is what is called *multi-factor authentication*, and the most practical method is two-factor authentication. For example, users need to connect a hardware device to a computer and their passwords to enable their access to an Internet banking system, this kind of authentication is two-factor authentication.

The rational of the above authentication mechanisms is as follows: something you know is a secret, others (apart from the verifier, where the verifier may be a person, a platform, or a database) do not have the secret, hence cannot forge the authentication. Something you have is based on the assumption that you can keep the physical thing in a safe place. This assumption is weak, since swipe cards can get lost. Something you are is based on the uniqueness of biometric information, assuming that others do not have similar biometric information that can pass the fuzzy matching. However, attackers may be able to read your biometric information, duplicate your biometric information, and hence can illegally pass the authentication. This brief analysis shows that, something you have and something you are do not provide high level of security, but for the purposes of convenience. For secure communications, something you know is the security basis.

There are different authentication protocols. The most commonly used method for authentication is called *challenge-response*, where the *challenge* is a message created by the verifier which is unpredictable, and the *response* is the answer that the prover provides to the verifier. The response should be something created using some secret information known only to the prover (and sometimes the verifier as well), and is verifiable.

To save the communication cost, it is possible to design non-interactive authentication protocols. Although the non-interactive authentication protocols do not have challenge-response process, the mechanism is similar to the challenge-response. Instead of using a challenge which is a random number, the non-interactive authentication protocols use something computed by the prover, and have some randomness and should not be manipulated by the prover.

In order to show how authentication protocols work, some basic authentication protocol are presented here.

5.5.2 Authentication Based on Shared Key

In an authentication protocol, there are two parties: a prover who claims to be someone, and tries to prove the claim, and a verifier who checks whether the prover's claim is true. It is assumed that the prover and the verifier share a secret key k. Naturally we assume that a symmetric key encryption algorithm E is available to both the prover and the verifier. Then an authentication protocol can be made as follows:

(1) The verifier generates a random number R, encrypts R using the shared key k: $c = E_k(R)$, and sends the ciphertext c to the prover.
(2) The prover decrypts the ciphertext c to get R, and sends R back to the verifier.
(3) The verifier checks whether the received R is the same as the one he generated before. If so, then the authentication succeeds, otherwise the authentication fails.

There are alternatives to the above protocol. For example, instead of sending the ciphertext of R to the prover, the verifier can simply sends R to the prover, and the prover encrypts R using their shared key k and sends back the ciphertext. The verifier encrypts R and see whether the result of the encryption is the same as the message received from the prover. Similarly, more variants of the protocol can be easily designed.

Since the verifier believes that no one else knows k, and is unable to forge the response, hence, a successful protocol confirms the prover to be genuine.

5.5.3 Password-Based Authentication Protocols

A password is a secret to memorize. Because random numbers are difficult to remember, passwords are often in the format not so random and easy to memorize. Hence, a password is usually weaker than a randomly generated secret of the same size. Many information systems and network-based applications use password-based authentication. For example, a password is required to login a computer system, to login a user account from a data center, or to get access to a bank account.

A proper implementation of password-based authentication should have a challenge-response process between a prover and a verifier. There are poor implementations of password-based authentication protocols, where user passwords are transmitted over public networks as plaintexts. Such implementations are vulnerable to network attacks, and even a passive eavesdropping attack would be able to get the user passwords. Computer systems use a hashing algorithm to convert a password and a little piece of text called "salt", the output of the hashing algorithm together with the "salt" is stored in a special password file. When someone wants to login, the system can find the "salt" according to the user account name, then the user enters a password, and the system joins the password and the

salt, then apply the hashing algorithm to get an output, and compares the output with the stored hash code. If they match, then user is permitted to access. This kind of password-based authentication method has been used in many information systems, and is less secure than the change-response mechanism.

In order to increase the security level of authentication, some systems adopt *multi-factor authentication* mechanism. Multi-factor authentication means that more than one authentication method is used. The most common such authentication is two-factor authentication. For example, a password and a fingerprint are needed to complete the authentication.

5.5.4 Public Key Based Authentication

A public key certificate contains the information about of user identity and a public key owned by the user, together with other relevant information. These messages are signed by a certification authority (CA). Although the signature of a public key certificate can be publicly verified, it only proves that the identity in the certificate is the owner of the public key in the certificate. This does not confirm that someone providing the public key certificate is the owner of the certificate.

Assume that a prover claims to be the owner of a public key pk (or the public key certificate), an authentication protocol can be designed, which is composed of the following steps:

(1) The verifier generates a random number R, encrypts R using the public key pk, and sends the ciphertext to the prover.
(2) The prover decrypts the ciphertext using his private key to get R, and sends R back to the verifier.
(3) The verifier checks if the received R is the same as the one he generated before. If so, then the authentication succeeds, otherwise the authentication fails.

There are variants to the above protocol. For example, instead of sending the ciphertext of R to the prover, the verifier can simply send R to the prover, and the prover creates a signature on R using his private key sk, and sends the signature to the verifier. The verifier verifies whether the received message is indeed a digital signature on R created by the owner of public key pk. Since the verifier believes that no one else knows the private key sk, the successful execution of the authentication protocol confirms that the other party is indeed the owner of the public key.

5.5.5 Schnorr Identity Authentication Protocol

From the above description it is seen that, authentication protocols can easily be setup, either based on a shared secret key, or based on the user's public key, which can be practically provided via a public key certificate, so that the correctness of

the public key can be verified. However, a practical authentication protocol might be much more complicated. A typical such example is Schnorr authentication protocol [6]. In 1991, Schnorr presented an authentication protocol based on a digital signature scheme.

Schnorr's protocol assumes the existence of a trusted third party, called key authentication center (KAC). By saying a trusted third party, it means that users in the protocol are assumed to trust KAC, and have got the public key of KAC. In the stage of protocol initialization, the KAC choose the following parameters:

- A large prime number p, which should be practically larger than 2^{1024}.
- A prime number q satisfying $q|(p-1)$, i.e., q is a factor of $p-1$, and should be fairly large, say $q > 2^{160}$.
- An element α whose multiplicative order modulo p is q, i.e., $\alpha^q \bmod p = 1$, and $\alpha^k \bmod p \neq 1$ for all k with $0 < k < q$.
- A security parameter t, satisfying that $2^{2t} < q$. Schnorr suggested that t should be no less than 72. Considering the rapid technological development, t can be even larger, say $t = 80$.
- A hash function $H(x)$ known to the users involved in the protocol.

Each user in the system generates a random number $s < q$ as his private key, and computes the corresponding public key $v = \alpha^{-s} \pmod p$. It is assumed that a digital signature algorithm $Sign_{KAC}$ for KAC and its corresponding verification algorithm Ver_{KAC} are available. The KAC may provide digital signatures on user's identity and public key, making it like a public key certificate.

With the above preparation, Schnorr's authentication protocol can be described as follows. Assume that a prover A wants to prove its identity to a verifier B, the following steps are executed between A and B.

(1) A chooses a random number k, $0 \leq k \leq q-1$, and computes $\gamma = \alpha^k \bmod p$.
(2) A sends ID_A, v, S and γ to B, where ID_A is A's identity, $v = \alpha^{-s} \pmod p$ is A's public key, S is KAC's signature showing the connection between ID_A and v.
(3) B verifies KAC's signature to confirm the correctness of ID_A and v. Then B chooses a random number r, $1 \leq r \leq 2^t$, and sends r to A.
(4) A computes $y = (k + sr) \bmod q$, and sends y to B.
(5) B verifies whether equality $\gamma = \alpha^y v^r \bmod p$ holds. If so, then B accepts A as owning the true ID_A.

The security of Schnorr's identity authentication protocol is based on the discrete logarithm problem. Like other discrete-logarithm based security protocols, if the discrete logarithm problem can be efficiently solved, then the above protocol is broken, i.e., attacks can forge anyone's identity to pass the authentication verification. However, so far the discrete logarithm problem is still a hard problem, and with the scale of p being 1024 bits long, to solve the discrete logarithm problem is computationally infeasible.

5.6 Key Management

From the above description it can be seen that, if a message is encrypted, then the receiver should have the correct key to decrypt. The decryption key can be a commonly shared symmetric key, or a private key in a public key cryptosystem. If a message is protected with message authentication code, then the receiver should have the key used in the MAC algorithm, so that the integrity of the message can be verified. In both of the cases, there should be a proper way to share a secret key with the message receiver. This is the key management problem.

5.6.1 Symmetric Key Management

In a symmetric key encryption, both the sender who encrypts the message and the receiver who decrypts the encrypted message should have a common key, so that they can establish secure communications.

Now the problem is how to let the sender and the receiver share a common key. Conventionally, there should be a separate secure channel used exclusively for key transmission, so that a common key (or a set of keys) can be securely shared between the sender and the receiver. This communication model can be depicted in Fig. 5.1.

A typical such channel is the way to send codebooks as used by military in early ages. Even if with digital communication technology, to set up a secure channel for key sharing is very costly. Moreover, in the scenario of multi-user communications with n users, if a unique key is shared between every two out of the n users, then it is easy to compute that, the number of shared keys is $n(n-1)/2$. This number increases rapidly with the increase of n. Taking into consideration the requirement that a secret key needs to be updated from time to time, the overhead cost to manage the secret keys to enable secure communications in multi-user environment is unacceptably high.

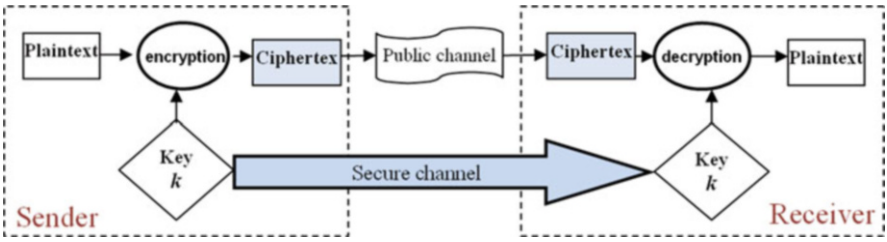

Fig. 5.1 The model of secure communication based on symmetric encryption

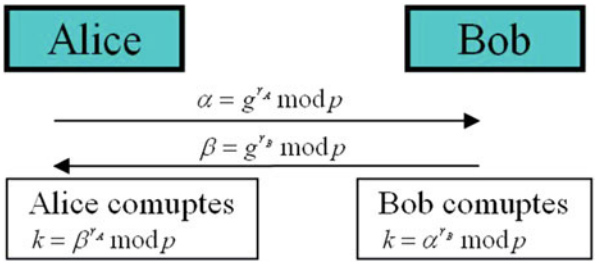

Fig. 5.2 Diffie-Hellman key agreement protocol

5.6.2 Diffie-Hellman Key Agreement Protocol

In 1976, Diffie and Hellman proposed an innovative method to enable secret key agreement over public channels. This method effectively solves the symmetric key management problems as mentioned above.

To setup the parameters to run the Diffie-Hellman key agreement protocol, two parties, Alice and Bob, agree on a common prime number p which should be sufficiently large, so that discrete logarithm problem is infeasible to solve. In addition, Alice and Bob agree on an integer g which is smaller than p. The protocol can be depicted in Fig. 5.2.

The process of Diffie-Hellman key agreement protocol can be described by the following steps:

(1) Alice chooses a random number $r_A < p$, computes $\alpha = g^{r_A} \bmod p$, and sends α to Bob;
(2) Bob chooses a random number $r_B < p$, computes $\beta = g^{r_B} \bmod p$, and sends β to Alice;
(3) When Alice receives β from Bob, Alice computes $k_1 = \beta^{r_A} \bmod p$;
(4) When Bob receives α from Alice, Bob computes $k_2 = \alpha^{r_B} \bmod p$.

It is easy to verify that $k_1 = k_2$, hence Alice and Bob agree on a key $k(= k_1 = k_2)$. Although the key agreement process is conducted over the public networks, no one else is able to recover their shared key.

It is noted that during the execution of the protocol, an opponent may be able to eavesdrop the transmitted data, including α and β. In addition, it is reasonable to assume that the opponent knows the public parameters p and g. However, the opponent does not know the randomly generated parameters r_A and r_B. The security of the protocol is about whether the opponent can compute the shared key k. More precisely, the security problem of Diffie-Hellman protocol is that, given p, g, $g^{r_1} \bmod p$, $g^{r_2} \bmod p$, how hard is it to compute $g^{r_1 r_2} \bmod p$. This problem is known as *Diffie-Hellman problem*, it is believed to be the same hard as discrete logarithm problem. The discrete logarithm problem can be stated as: given g, p as in Diffie-Hellman protocol, and given an arbitrary value $y = g^x \bmod p$, to

compute x, where x and y are all integers. Without modulo operation, this is a conventional logarithm problem that can easily be solved. However, the discrete logarithm problem is believed to be computationally infeasible to solve, if p is sufficiently large.

From the above description it can be seen that, the Diffie-Hellman key agreement protocol provides a way to allow Alice and Bob to securely setup a common key k, and others do not know k, even though they may have monitored the whole execution process of the protocol carefully.

5.6.3 Man-in-the-Middle Attack on Diffie-Hellman Protocol

As mentioned above, Diffie-Hellman key agreement protocol is secure against eavesdropping attack. However, under active attacks, the protocol is not secure. Assume that in the process of Diffie-Hellman key agreement protocol, when Alice and Bob are about to agree on a common secret key, an attacker Eve in between Alice and Bob intercepts the data transmission between Alice and Bob, and replaces the message by her own. More specifically, when Alice sends $\alpha = g^{r_A} \bmod p$ to Bob, Eve intercepts it, and sends $\gamma = g^{r_E} \bmod p$ to Bob instead of α, where r_E is a random number generated by Eve. Similarly, when Bob sends $\beta = g^{r_B} \bmod p$ to Alice, Eve intercepts it, and sends $\gamma = g^{r_E} \bmod p$ to Alice instead of β. By following the steps of Diffie-Hellman key agreement protocol, Alice would compute $k_1 = \gamma^{r_A} \bmod p = g^{r_A r_E} \bmod p$, and Bob would compute $k_2 = \gamma^{r_B} \bmod p = g^{r_B r_E} \bmod p$. Since $k_1 \neq k_2$, Alice and Bob does not have a common key. However, Eve computes

$$k_1 = \alpha^{r_E} \bmod p = g^{r_A r_E} \bmod p,$$
$$k_2 = \beta^{r_E} \bmod p = g^{r_B r_E} \bmod p.$$

Hence, Eve shares a secret key k_1 with Alice and another secret key k_2 with Bob. When Alice and Bob use their keys to communicate, Eve intercepts the messages from Alice, decrypts them using key k_1, then encrypts the result using key k_2 and sends to Bob; In the mean time, Eve intercepts the messages from Bob, decrypts them using key k_2, then encrypts the result using key k_1 and sends to Alice. By doing that, both Alice and Bob feel that they could successfully communicate using their agreed key, as if it were a secure communication. The feature of this attack is that Eve sits in between Alice and Bob and intercepts all their communications and makes proper replacement. This kind of attack is called *man-in-the-middle attack*. In general, man-in-the-middle attack is such that a third-party intercepts a communication between users, gets the message, modifies the message, and then sends to the receiver. This is similar to message integrity attack, however, its target is not simply modify a little part of the message, but to replace the intercepted message

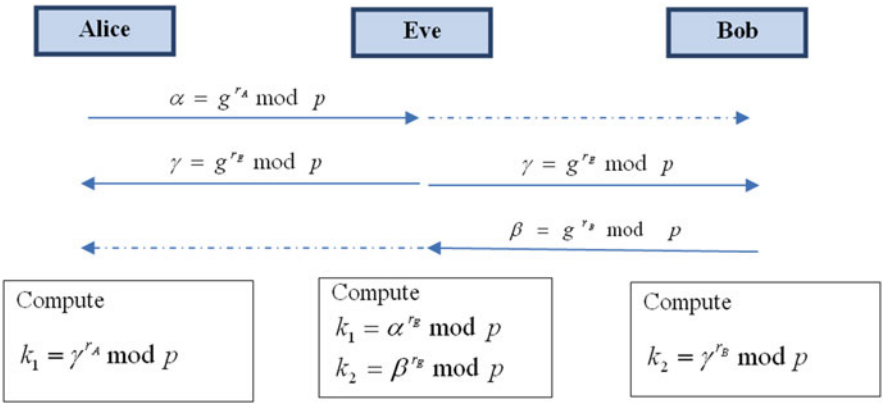

Fig. 5.3 The process of man-in-the-middle attack on Diffie-Hellman key agreement protocol

with a new one. The process of man-in-the-middle attack on Diffie-Hellman key agreement protocol can be shown in Fig. 5.3.

It can be seen that, the existence of the man-in-the-middle attack on Diffie-Hellman key agreement protocol is due to lack of authentication, i.e., when someone claims to be Alice, Bob believes that it was Alice, and when someone claims to be Bob, Alice also believes that it was Bob. By adding the authentication service, Diffie-Hellman key agreement protocol can be modified to be resistant against man-in-the-middle attack.

5.6.4 Symmetric Key Distribution Using a Public Key Cryptosystem

Section 5.6.1 describes the symmetric key management problem, and Diffie-Hellman key agreement protocol as described in Sect. 5.6.2 presents a method to allow two users to agree on a secret key over public networks.

In fact, with the help of public key cryptography, symmetric key management can be made even simpler: Alice generates a random number as a secret key k, Alice encrypts k using Bob's public key, and sends the ciphertext to Bob. When Bob receives the message, he decrypts the ciphertext using his private key to get the secret key k. Now Alice and Bob have a common secret key k, and no one else is able to get k. In fact, this is the model that contemporary secure communication uses. Practically, Alice does not need to agree on a key with Bob in advance, the secret key can be shared at the same time when secure data transmission is needed.

More precisely, when Alice wants to securely send a message m to Bob, she does the following:

(1) Generates a random number as a secret key k, encrypts k using Bob's public key to get ciphertext c_1;
(2) Encrypts the message m using a symmetric key encryption algorithm with k as the encryption key, and gets ciphertext c_2;
(3) Sends c_1 and c_2 together to Bob.

When Bob receives the ciphertexts, he does the following:

(1) Decrypts c_1 using his private key to get the secret key k;
(2) Decrypts c_2 using k and the symmetric decryption algorithm to get m.

In this manner, the convenience of using public key cryptosystem and the efficiency of using symmetric encryption are effectively combined together.

As is shown in Fig. 5.3, Diffie-Hellman key agreement protocol is vulnerable to man-in-the-middle attack. Using a public key encryption to transmit a secrete key has the same problem. Let Eve be an attacker who can intercept all the communications between Alice and Bob, and can make modifications/replacements at discretion. Then the following steps can be used by Eve to conduct an impersonation attack or even a man-in-the-middle attack.

(1) Alice sends a message to Bob asking Bob's public key.
(2) Eve intercepts the message, sends her own key pk^E to Alice, claiming that it is Bob's public key.
(3) Alice chooses a random number k, uses pk_E to encrypt k to get c_1; Alice encrypts the message m using k and a symmetric key encryption algorithm to get c_2, and sends c_1 and c_2 to Bob;
(4) Eve intercepts the message c_1 and c_2, uses her own private key to decrypt c_1 to get k, then uses k to decrypt c_2 to get the message m.

When the above steps are completed, Alice believes that she has securely sent message m to Bob. However, the actual receiver is Eve. The above is a simple impersonation attack. If further communication from Bob is needed, and the responses from Bob cannot be manipulated by Eve, then Eve can lunch a man-in-the-middle attack, which is a little more complicated than the impersonation attack. Like the vulnerability of Diffie-Hellman key agreement protocol, the causes of impersonation attack and man-in-the-middle attack are due to lake of authentication.

5.7 Introduction of PKI

As described above, an efficient way to share a secret key between communication parties is to establish such a common key over public channels (e.g., using Diffie-Hellman key agreement protocol), or to let one party create a random number as

a session key and encrypt the session key using the other party's public key and transmit it to the other party over the public networks.

However, the public key being used may not be the correct one. For example, if Eve provides her public key claiming to be Bob's public key, then Alice's message may be obtained by Eve instead of Bob. Eve can forge Bob the same way. So, there must be a way to confirm that the pubic key supposed to belong to someone is really the case. This can be done by employing a trusted third party (TTP), who acts as an authority, creates a digital signature on a message that links the identity and its public key, which is called the *public key certificate*. The TTP can issue a public key certificate to all authorized users. In this case, Alice can verify the correctness of Bob's public key certificate. However, this verification only confirms that the public key in the certificate belongs to Bob, and cannot be sure that the party communicating with her is really Bob. Fortunately, with the help of public key certificates, impersonation attack and man-in-the-middle attack do not work, because identity impersonation can be detected by the authentication process.

5.7.1 What is Public Key Certificate

Assume that Alice and Bob do not trust each other, and want to establish secure communication. If they both trust a third party TTP, then they can ask the TTP to provide their public keys, i.e., to send Alice's public key to Bob, and send Bob's public key to Alice. However, when the TTP serves a large number of users, the efficiency of this model encounters a big challenge. Note that an effective way for the TTP to confirm Bob's public key to Alice is to digitally sign a message saying that a specific public key belongs to Bob, this signed message can be made well in advance before Alice or anyone else makes such a request. Similarly, the TTP can digitally sign a message for each user confirming the linkage between a public key and the identity of the user. In order to make the signed message more formal, and considering that there may be more than one TTP, the messages singed by the TTP should contain the following information in addition to the public key and the identity of the user:

(1) The identity of the TTP;
(2) The validity of the certificate;
(3) The purpose of the certificate, i.e., what the certificate is used for.

It can be seen that, once such public key certificates are generated, there are a series of problems: where to put the certificates? How does a user get a specific public certificate? What if a user wants to abolish his/her public key? All such problems lead to the need of systematical management of public key certificates.

The system to manage and use public key certificates is called *public key infrastructure (PKI)*, which includes how the public key certificates are created, stored, verified, revoked, etc.. In practical applications, a PKI system is composed of hardware, software, management policies, and formal protocols. In a PKI system,

there is an authority called *certification authority (CA)* who is in charge of issuing public key certificates. CA has the following responsibilities:

(1) Issue public key certificates for all qualified and authorized users.
(2) When a public key certificate is to be abolished within its validity period, to assign a message claiming the certificate to be revoked, and the message is added into the *certificate revocation list (CRL)*.
(3) Put the valid public key certificates and the CRL into a specific database, keep the database updated in time whenever new public key certificates are issued or the CRL is updated.

For a large PKI system, apart from the CA, there needs to have other authorities assisting to run the PKI system. One type of such authorities are the *registration authorities (RA's)*. The RA's can help to improve the efficiency of running the PKI system when the number of users is very large. The responsibilities of an RA include the following:

(1) When a user applies for a new public key certificate, verify the authenticity and qualification of the user.
(2) When the CA issues new public key certificates, verify the validity of the certificates before they are sent to the users or to the public key certificate database.

The most well known public key infrastructure is the X.509 standard series published by ITU-T [7]. The public key certificates based on the X.509 system have widely been used in web browsers.

5.7.2 The X.509 Public Key Certificates

In 1988, the Telecommunication Standardization Sector of ITU (International Telecommunication Union) leaded the X.509 standard, which forms a standard document under the X.500 series of standards, and has become an IOS/IEC standard. In 1997, the X.509v3 was published, which adds some extended fields in public key certificates. The specification was revised in 2000, the revision further clarifies the roles of CRL and certificate attributes.

Noticing that the specification of X.509 is suitable for the Internet, the Internet Engineering Task Force (IETF) adopted it as RFC 2459 [2], and later on it was updated as RFC 3280 [3].

As categorized by X.509, there are different kinds of public key certificates, including general-purpose public key certificates, attribute certificates, and qualified certificates. A general-purpose public key certificate proves the linkage between a specific public key and its owner; an attribute certificate also shows the rights of the public key owner; and a qualified certificate is issued to a human user for the purpose of digital signature verification. If a public key certificate does not have specific indication, then it is a general-purpose public key certificate.

An X.509 public key certificate has many fields, including version, serial number, validity, issuer ID, owner ID, public key information, signature algorithm ID, signature value, and extension, where the validity includes the starting time when the certificate becomes valid, and the ending time when the certificate becomes invalid.

The document of X.509 standard specifies a number of processes in using the public key certificates, including how the certificates are issued, where they are stored, how to revoke and update a public key certificate, how to obtain a public key certificate, and how to verify the validity of a public key certificate.

5.7.3 The Trust Relations When There Is More than One CA

The use of public key certificates assumes that all the users trust the same CA. This is impractical when the system becomes very large, particularly when the communication crosses organizations, regions, or even countries. In this case, in order to enable the establishment of trust among different regions that are managed by different CAs, a mechanism to link different CAs is necessary.

There are different CA structures that can be used to establish the relations of trust between different CAs, some of the typical CA structures for the relation of trust are as follows:

Top-down CA-tree structure: Assume that there is a root CA, denoted as CA_0, who issues public key certificates to some sub-CAs, say CA_1, CA_2, $\cdots$, CA_n, where those sub-CAs are assumed to trust the root CA. Then each sub-CA issues public key certificates to sub-sub-CAs. For example, CA_1 issues public key certificates to $CA_{11}, CA_{12}, \cdots$, CA_2 issues public key certificates to $CA_{21}, CA_{22}, \cdots$. This process can continue to multiple levels. At certain stage, a CA at certain level issues public key certificates to lower level sub-CAs, and/or end users. An example of the top-down CA-tree structure is shown in Fig. 5.4.
With the top-down structure, every node (a CA or an end user) trusts all the ancestors and hence has their public keys. For example, in Fig. 5.4, user A has the public keys of CA_{11}, CA_1, and CA_0, while CA_{21} has the public keys of CA_2 and CA_0. With this setup, end users can always find a path of trust, so that their public key certificates can be verified even crossing different CAs. For example, as in Fig. 5.4, when user A wants to verify user C's public key certificate, the path of trust is $A - CA_1 - CA_{12} - C$, this means that, A has CA_1's public key, so A can verify the validity of CA_{12}'s public key certificate issued by CA_1. When this verification succeeds, A can use CA_{12}'s public key to verify the validity of C's public key certificate. Similarly, if user C wants to verify the public key certificate of user E, then the path of trust is $C - CA_0 - CA_2 - CA_{21} - E$, and the process of verification is similar.

Two-way CA-tree structure: In the top-down CA-tree structure, each end user and intermediate CA has to trust one or more CAs. Since the setup of initial trust is usually done in a trusted environment (e.g., off-line), which costs more than

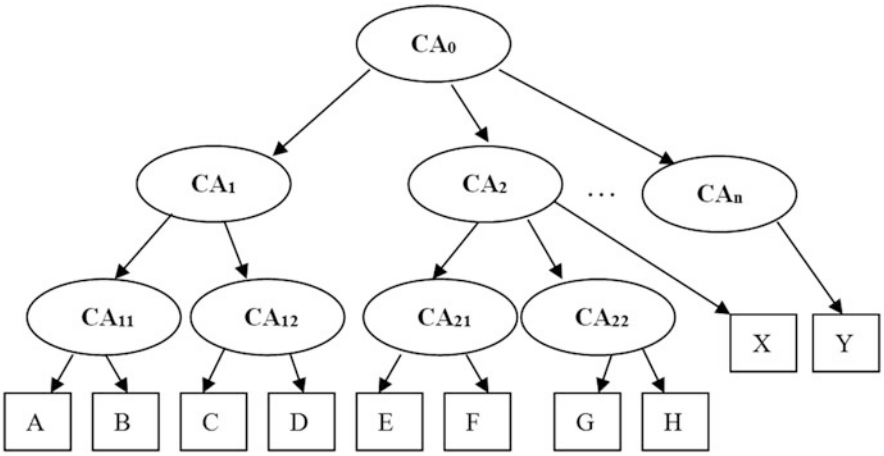

Fig. 5.4 An example of the top-down CA-tree structure

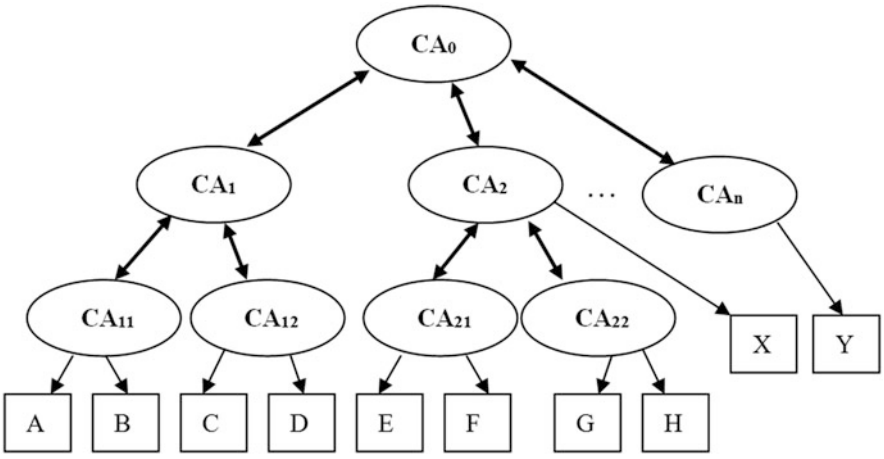

Fig. 5.5 An example of the two-way CA-tree structure

to establish a trust based on existing relations, it would be a good saving if the number of initial trust relations can be reduced. This seems difficult, however, an alternative is to set up mutual trust between the related CAs, so that each CA only needs to set up the initial trust with its ancestor and descendant CAs, and each end user only need to set up the initial trust with the nearest CA. An example of this structure can be shown in Fig. 5.5.

With the two-way CA-tree structure, end users can also find a path of trust, so that their public key certificates can be verified crossing different CAs. For example, as in Fig. 5.5, it can be seen that the path of trust from user A to user C is $A - CA_{11} - CA_1 - CA_{12} - C$, this means that, A trusts CA_{11},

hence has CA_{11}'s public key, which can be used to validate CA_1's public key certificate issued by CA_{11}, hence can confirm CA_1's public key, which can be used to validate CA_{12}'s public key certificate issued by CA_1, hence can confirm CA_{12}'s public key, which is then used to verify the validity of user C's public key certificate, hence can establish the relation of trust on C. Similarly, if user C wants to establish the relation of trust on user E, then the path of trust is $C - CA_{12} - CA_1 - CA_0 - CA_2 - CA_{21} - E$, and the process of verification is similar.

Although the path of trust in the two-way CA-tree structure may be longer than that in the top-down CA-tree structure, in the two-way CA-tree structure, users only need to trust their closest CA (i.e., to record one CA's public key certificate), where in the top-down CA-tree structure, users have to trust all the ancestor CAs.

Bridge-CA structure: In the above structures, a root CA is assumed. However, in practical applications, there is a need to connect two or more independent CA-trees, enabling the paths of trust to be established crossing the two CA-trees. In this case, it is impractical to find another party to act as their ancestor CA. The problem can be solved by establishing a mutual trust between two top-level CAs, which looks like a bridge connecting the two structures together. Assuming that CA_1 is a root CA in a CA-tree, and CA_2 is a root CA in another CA-tree, by establishing a mutual relation of trust, then users crossing the two CA-trees can validate their public key certificates, hence can establish the relation of trust. This structure is called *bridge-CA structure*. An example of such structure is shown in Fig. 5.6.

The advantage of the bridge-CA structure is that, two independent CA structures can establish trust connection at anytime, and the internal relation of trust in the two CA structures can differ. This gives a lot of flexibility.

Based on the assumption of initial trust, more relations of trust can be established among end users. When the relation of trust is established, authentication can be done effectively and with trust.

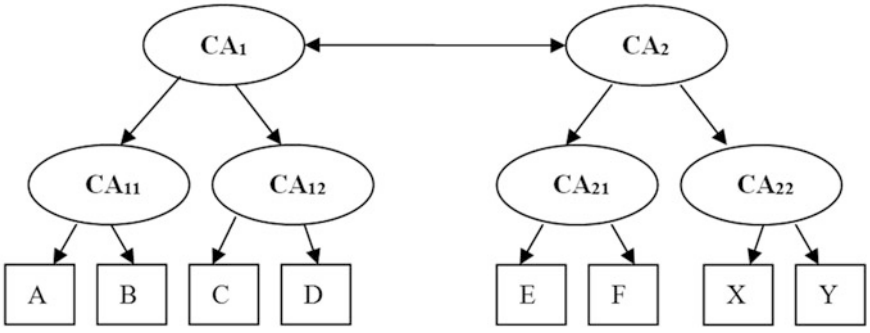

Fig. 5.6 An example of the bridge-CA structure

Table 5.1 The comparison between X.509 certificates and WTLS certificates

X.509 certificates	WTLS certificates
Version	Version
Serial number	
Algorithm identifier	Algorithm identifier
Name of issuer	Name of issuer
Period of validity	Period of validity
Subject (owner)	Subject (owner)
Subject's public key	Subject's public key
Issuer ID	
Subject ID	
Signature algorithm	Signature algorithm
Issuer's signature	Issuer's signature
Extension (optional)	

5.8 Lightweight PKI for IoT Applications

Public key certificates are made for traditional information systems with Internet connections. However, many IoT devices that connected to the Internet have limited resources, and using a normal X.509 public key certificate consumes much of resources. In order to facilitate the usage of IoT devices, given that many fields of X.509 public key certificates are useless for IOT devices, there is a demand to design a simple version of public key certificate. Based on X.509 public key certificates, a simplified version targeting at wireless communication has been designed, which is the WTLS certificates. The WTLS certificates are more suitable to IoT devices than the normal X.509 certificates due to its simplicity. However, they are only relatively simpler than X.509 certificates. The comparison of X.509 certificates and WTLS certificates can be seen from Table 5.1.

It can be seen from Table 5.1 that, some of the fields in X.509 certificates are missing in the WTLS certificates, this simplification is meant to reduce the size of the WTLS certificates. However, although the WTLS certificates are smaller in size than X.509 certificate, they are still quite large. For IoT devices with limited resources, public key encryptions should be avoided wherever possible.

5.9 Summary

This chapter introduces the concept and basic mechanisms of the relation of trust. It is shown that the relation of trust cannot be created from scratch, it can be extended from existing relation of trust, hence there must be some initial relation of trust to start with. The initial relation of trust can be established by a pre-shared secret key, which exists before any secure communication is conducted, or by public key certificates which are written into IoT devices by their manufacturers or operators.

This chapter also describes some typical key management protocols, discusses some security problems of such protocols under the condition with and without the establishment of trust.

It is shown that by using a public key encryption, a secret can easily be shared between two communication parties. To ensure that the communication parties are not forged by a wrong public key, the authenticity of a public key can be verified using a public key certificate, and the verification of the certificate is based on the assumption that the verifier trusts on the certificate issuing authority. The X.509 public key certificates are widely used.

References

1. C. Everett, Cloud computing: a question of trust. Comput. Fraud Security **2009**(9), 5–7 (2009)
2. R. Housley, et. al., Internet X.509 Public Key Infrastructure Certificate and CRL Profile, The Internet Society (1999). https://tools.ietf.org/html/rfc2459
3. R. Housley, et. al., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, The Internet Society (2002). https://tools.ietf.org/html/rfc3280
4. J. Huang, D.M. Nicol, Trust mechanisms for cloud computing. J. Cloud Comput. **2**, Article number: 9 (2013). https://doi.org/10.1186/2192-113X-2-9
5. K. Khan, Q. Malluhi. Establishing trust in cloud computing. IT Professional **12**(5), 20–27 (2010)
6. C.P. Schnorr, Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991)
7. X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. International Telecommunication Union (2008). https://www.itu.int/rec/T-REC-X.509
8. Y. Zhang, X. Xu, A. Liu, Q. Lu, L. Xu, F. Tao, Blockchain-based trust mechanism for IoT-based smart manufacturing system. IEEE Trans. Comput. Soc. Syst. **6**(6), 1386–1394 (2009). https://doi.org/10.1109/TCSS.2019.2918467

Chapter 6
IoT Perception Layer Security

The IoT perception layer is where digital data are connected to physical devices. Although many IoT devices have sufficient computation and communication capabilities, the resource-constraint feature of some IoT devices lead to different security requirements and security techniques. Some security threats in IoT perception layer are introduced, including eavesdropping attack which has little cost in wireless communication networks, traffic analysis attack which does not need to recover the content of the communication data, impersonation attack and data modification attack as in traditional network environment, laboratory analysis which is specific to chips, and applies to IoT devices that are easy to capture and disassemble, cloning attack which is mainly on RFID tags, sybil attack which was proposed for wireless sensor networks and applies to some IoT applications, energy exhaustion attack which applies to battery powered devices, replay attack which may cause severe problems when the replayed data is a control command, regardless whether the data has encryption and/or integrity protection, and botnet control attack which makes use of large amount of IoT devices. Some possible countermeasures against those attacks are briefly described, and some advices about IoT perception layer security mechanism are proposed.

6.1 Introduction

From the IoT architecture (see Fig. 2.1) it can be seen that, the processing layer is a platform for data processing, the network layer is a tunnel for data transmission. Both of these layers deal with data only. However, the perception layer contains sensors and controllers/actuators. Although the sensors are used to collect environmental data, the controllers/actuators can be used to control, to justify, or to operate physical devices. This connection from information (data) to physical devices is a special feature for IoT security services to consider. Hence, security protection

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_6

for the perception layer should include data security and security protection on operation of physical devices.

However, traditional security techniques can hardly be implanted directly in the IoT perception layer, because the perception layer has a large variety of devices, and their performance and capabilities differ significantly. Apart from the cases where the IoT devices are similar to traditional information processing devices, some special features of certain IoT devices are as follows:

1. Many IoT devices are put in an open environment without surveillance, they face the security threats of physical damage and network attack. Attackers may be able to intrude those devices by physical connection or wireless tampering.
2. The wide application of wireless communication by IoT devices reduces the difficulty of eavesdropping attack. Moreover, attackers may lunch replay attack, which may cause unexpected damage if the replayed message is a control command.
3. Many of the IoT devices have constrained resources, including the capability of storage, computation, and communication. There are technical challenges to use security techniques in such devices. Hence, lightweight security techniques become a specific requirement in many IoT applications.
4. In practical implementations, in order to reduce the implementation cost, many low cost IoT devices share a same secret key. In this case, once an attacker captures one such device and gets the secret key by laboratory analysis, other IoT devices sharing a same key then have no more security. Note that laboratory analysis has higher threats than the security analysis based on data only. Hence, sharing a secret key among multiple IoT devices introduces security vulnerability, and should be avoided as much as possible.
5. Attackers may have little interest in getting information from the IoT devices, but to control them to form a botnet, and lunch DDoS attacks on other systems.

The special features of IoT devices make IoT security mechanisms different from those for traditional information systems.

6.2 Security Threats in IoT Perception Layer

Given the special features of IoT devices, the attacks on the IoT perception layer are often different from those on traditional information systems. Nevertheless, large amount of traditional attacks on information systems will affect the IoT perception layer. The basic traditional security measures are confidentiality, integrity and availability, where the confidentiality is about protecting data content from illegal stealing, integrity is about detecting possible illegal modification of data, and availability is meant to ensure services to be provided as they are expected. In the IoT perception layer, these security measures are still applicable. However, many security techniques used in the IoT perception layer are significantly different from those in traditional information systems.

6.2.1 Security Threats and Countermeasures Against Eavesdropping Attack

Eavesdropping is a common security threat in network environment, particularly in the environment of wireless networks. In the perception layer of IoT, large amount of communication is done over the wireless networks, and this seems to be a development trend in the IoT perception layer.

Fortunately, there are mature techniques that can be used to effectively protect eavesdropping attacks. A simple encryption will do. For example, assume that the original message to be transmitted is m, using an encryption algorithm, with the involvement of a secret key k, the plaintext message m can be transformed into a ciphertext $c = E_k(m)$. Although the activity of eavesdropping cannot be avoided, however, the eavesdropping can only get the ciphertext c, and the attacker cannot recover the original message m without knowing the key k, this is the security measure of encryption algorithms. It is noted that, the implementation of an encryption/decryption algorithm requires some hardware and/or software resources, some low cost IoT devices may not have sufficient resources to implement traditional encryption algorithms, hence lightweight encryption algorithms have been designed, and a number of such algorithms have become available (e.g., [1, 4–6]).

6.2.2 Security Threats and Countermeasures Against Traffic Analysis Attack

Traffic analysis attack [3] is a traditional security threat. When the communication data are encrypted, eavesdropping attack does not work, because attackers are unable to recover the original message from the captured data. However, by analyzing the data traffic, some important information/implementation may be revealed.

Traffic analysis attack is based on what the attacker observes in the network. This type of attack does not need to break the encryption. From traffic analysis, an attacker can find out the communication addresses, the routing structure, and even application behavior patterns.

In Internet applications such as voice over IP (VoIP), a useful protection mechanism against traffic analysis is to encrypt the traffic of Session Initiation Protocol (SIP). The use of a proxy server can force communication traffic to be concentrated to the proxy, which also helps to prevent traffic analysis attacks. In some network topologies, this can be achieved using a network address translation (NAT). The use of virtual private network (VPN) can achieve certain level of countermeasure against traffic analysis attack.

While hiding the real communication addresses can mitigate the traffic analysis attacks, this is not an ideal solution to IoT applications, because the attackers can get the data from the very beginning of the original sender before the data reaches any

proxy or server. A typical case is electricity meter reading, attackers can collect the meter reading data from the source of each meter. If the data is encrypted, the attackers can observe the size of the encrypted data to judge whether the electricity consumption of a household is normal (which means that the household is occupied), or very little electricity consumption (which means that the household is likely to be unoccupied). In order to thwart this kind of traffic analysis attacks, an effective technique is to balance the data size in communication, i.e., to change a small size message into the same size as a large one by padding some redundant message, so that the size of ciphertexts look all the same, so that the size of ciphertexts has no connection with the size of the original plaintext messages.

6.2.3 Security Threats and Countermeasures Against Impersonation Attack

If an IoT application simply checks the data format (e.g., a temperature sensor data) and ignore the data source, then attackers can easily create fake data misleading the data-based observation and analysis. Even if the identities of data source are fixed, e.g., the identities are stored in a database, an attacker can impersonate an authorized identity to send forged data.

The way to prevent impersonation attack is to use identity authentication mechanism, which is to ensure whether the claimed identity is the true identity of the communication party. Fortunately, many IoT applications use authentication mechanism to ensure the authenticity of identities, preventing connection attempts from impersonated identities. More detailed process of identity authentication can be found in Sect. 5.5.

It should be noted that, identity authentication is usually associated with key agreement, which means that the communication data are likely encrypted and/or with integrity protection.

6.2.4 Security Threats and Countermeasures Against Data Modification Attack

Data forgery attack is to illegally create some forged data aiming to cause confusion or misleading to the receiver. However, by using an encryption algorithm, data forgery attack can effectively be prevented.

Data modification attack is like data forgery attack, but not to forge data from scratch. Data modification attack is to modify the data from authentic communications, and the amount of modification is often very small. However, small modification may cause big change on the data. Noticing that the majority means of data transmission in IoT perception layer uses wireless communication, where

the most appropriate encryption mechanism is byte-based encryption, which can be achieved by using a stream cipher or a block cipher in CTR mode (or CFB mode, or OFB mode). If the attacker knows exactly the meaning of each bit of communication data, then by flipping a few bits (e.g., replace 0 with 1 and replace 1 with 0), the modified data can be significantly different from the original after decryption. It is known that modification attack can be effectively prevented by data integrity service, where the most common and effective method is to use a message authentication code (MAC), and a secret key is needed in order to make the MAC algorithm work. More description about the MAC can be found in Sect. 4.4.2 of Chap. 4.

There is another kind of forgery attack, called *cross-site forgery attack*. This kind of forgery attack works in Web based applications, which is in processing/application layer, and has different mechanism with the forgery attack as described above.

The above security threats are similar to traditional network attacks, except that the protection methods are different in the IoT perception layer. Apart from those security threats, there are some attacks that are particularly effective to the IoT perception layer, such attacks include laboratory analysis, RFID tag cloning, Sybil attack, energy exhaustion attack, replay attack, and botnet control.

6.2.5 *Security Threats and Countermeasures Against Laboratory Analysis*

The laboratory analysis means that a device is being analyzed in a laboratory, where some advanced tools are available, such as emission scan and hardware reverse engineering. The rationale for this kind of attacks is that many IoT devices are easy to capture and bring to a laboratory. If many IoT devices share a same secret key, then a successful laboratory analysis on one device can break the security of the whole IoT application system.

It is noted that, to prevent IoT devices from being stolen is impractical. Hence the mechanism of countermeasures against laboratory analysis is to increase the difficulty of laboratory analysis. Relevant such techniques include chip packaging, anti-reverse engineering, and side-channel protection. Those techniques obviously increase the cost of chips as the core processing units in IoT devices.

6.2.6 *Security Threats and Countermeasures Against Cloning Attack*

Cloning IoT devices can be cheap, and the security threat depends on the application of the IoT devices. For example, cloning RFID tags can cause severe security

problems. If such an RFID tag is used as ETC kind of fee collection, then the expenditure of the cloned device is credited to the owner of the original RFID tag. If the RFID tag is used as an entry access, then the cloned tag could be maliciously used by unauthorized access to a physical place, which should be strictly prohibited.

Even if the IoT devices have security protection, a security flaw may lead to the devices being cloned. For example, the earlier version of SIM cards in GSM communication system could be cloned easily.

Since the process of cloning can be done in a laboratory, and the one to be cloned can be under laboratory analysis, the security mechanism to thwart cloning attack should include two aspects: (1) The devices (e.g., RFID tags) should have security protection, and a secret key has been embedded inside the devices; (2) Some mechanisms to protect laboratory analysis should be used (e.g., the use of security chips).

6.2.7 Security Threats and Countermeasures Against Sybil Attack

In IoT perception layer, sensors are often used to collect environmental data. In order to get reliable data even in the case of a faulty sensor, sensors are often used with redundancy. However, if a forged sensor can represent multiple identities and send forged data, then a substantial fraction of data is forged, which gives a wrong picture of the environmental information. This is a typical "Sybil attack".

The concept of Sybil attack was proposed by Douceur in 2002 [2]. This attack applies to IoT devices, and the implementation of a Sybil attack is usually in the perception layer.

To thwart Sybil attacks, identity authentication can work effectively. By certifying each authorized identity a unique secret key, and the process of identity authentication requires the key, then the simulated virtue devices either have no authorized identities, or cannot pass the identity authentication process if they impersonate other authorized identities.

However, in the case where all the IoT devices share a same secret key, which is a common practice, identity authentication is often replaced by correct format of message encryption. In this case, since the identity information is public, the Sybil attack can be successful, and can forge as many identities as possible. Even in such case, it is possible to find the possibility of attacking activity, if the data processing center finds that the data coming from a same identity have abnormal changes.

It should be noted that, the Sybil attack is not always a bad thing, the mechanism of the Sybil attack can be used in location privacy protection.

6.2.8 Security Threats and Countermeasures Against Energy Exhaustion Attack

Large amount of IoT devices use batteries to supply power. When the batteries get flat, then the devices would stop working. Some IoT devices use removable batteries, and when the batteries get flat, the devices can be back to work once the batteries are replaced. There are some IoT devices that are too small to have removable batteries, or not worth to have removable batters due to the manufacturing cost. The embedded batteries of such IoT devices are supposed to last the whole lifetime. For such devices, the flat batteries mean the death of the devices.

Energy exhaustion attack is to excessively consume the energy of IoT devices, aiming to exhaust the batteries quickly. Energy exhaustion attack is effective even though the IoT devices have security protection mechanisms such as identity authentication and data encryption, because even unsuccessful connection attempts consume the energy of the devices.

The effect of energy exhaustion attack is like a DoS attack, hence is considered as a special DoS attack to resource-constrained IoT devices. The energy exhaustion attack is fatal to IoT devices with integrated batteries, because such attacks can cause them to die quickly.

Although the effect of energy exhaustion attack is like DoS attack, the resource-constrained IoT devices do not have capabilities to thwart such attacks. However, a properly designed dormancy mechanism can be an effect way to mitigate the effect of such attacks. This mechanism seems to be passive but effective. For example, a meter reading device is required to send the meter reading data a number of times a day, and sending the data requires a few seconds or even less. Why stay online all the time? Such a device can be set to dormancy most of the time. However, there may be cases that the data center asks for meter reading data at any time, in this case, dormancy can still be made possible. Assume that a meter-reading device sleeps for 59 s and then wakes up listening to the network for a second, if no special command is received, then sleep for another 59 s till next time of waking up. Given the fact that in most of the time, the devices are in the state of dormancy, the working time is about 1/60, hence this dormancy mechanism saves much energy. If an energy exhaustion attack is lunched, the device finds that the message from the attack is illegal, then goes into sleep, avoiding 59 s of being affected.

If a command is received, or it is time to send meter reading data, then it just needs to finish the task before going into the next period of dormancy. When the data processing center sends a command asking the device to do something, in order to ensure that the command can be received, the data processing center needs to send the command at least 2 times a second, and keep sending the command for 60 s, so that at least one message falls into a time slot of device is wakened.

6.2.9 Security Threats and Countermeasures Against Replay Attack

Although the concept of replay attack was proposed in traditional network environment, the effect of replay attack is not a serious threat on Internet. However, for IoT applications, replay attack can be a severe threat, because there are important control commands transmitted over the network, successful replaying such commands means that the commands may be executed at a wrong time, which cause serious problems.

The most effective countermeasure against replay attack is to label whether a message is fresh. Practically, a timestamp, or a counter, can be used for this purpose. For example, if an IoT device has a clock (CPU clock or RTC), let T be a timestamp of current time. When sending a message m, the data to be sent can be in the format

$$T \| E_k(T \| m),$$

where k is an encryption key, $\|$ is the concatenation operation. When the message is received by the data processing center, it checks the validity of T, i.e., whether the difference between the system clock time and T is within an allowable range, and decrypts the ciphertext part to recover $T \| m$. The purpose of putting T as a plaintext is for easy verification, and the purpose of putting T into encryption is to protect it from illegal modification. This method can prevent replay attack if a message is beyond the allowable range of time. The message freshness protection from the data processing center can be done similarly.

If the IoT device does not have any clock, then a counter can be used to provide evidence of message freshness. Let Ctr be a counter defined by an IoT device, then every time it sends a message m, it sends the following message:

$$Ctr \| E_k(Ctr \| m),$$

and then increments the counter, i.e., set $Ctr := Ctr + 1$.

On receiving the message, the data processing center does the following:

1. Check the validity of Ctr, i.e., check whether the value of Ctr is larger than a local counter record Ctr';
2. Decrypt the ciphertext part and get $Ctr \| m$, and check whether Ctr matches the plaintext part;
3. Accept m as a valid message;
4. Update the local counter value, i.e., set $Ctr' := Ctr$.

When the data processing center sends a message to the endpoint IoT device, the mechanism of message freshness protection is similar, but to use a different counter. This means that, a counter for incoming messages of party A matches a counter for outgoing messages of party B, and vice versa. It is noted that the mechanism of putting T or Ctr both in plaintext and in encryption provides data integrity service:

any illegal modification on the encrypted part of data and/or on T/Ctr will result in that the decrypted part of T/Ctr does not match that in plaintext, hence can be detected.

The advantages and disadvantages of using a system clock and counters for message freshness services are clear. The advantages of using a clock is the convenience, no need to keep record of the timestamps, but the disadvantages include the possibility of replay attack within the allowable time range, and the need of clock synchronization. While the advantage of using counters is the complete protection of replay attack, and the disadvantage is to keep a pair of counters for any direction of message transmission, and the counters have to be updated after every successful message transmission/reception.

6.2.10 Security Threats and Countermeasures Against Botnet Control

Due to the resource-constraint and limited resources, security protection of many IoT devices is also very limited or even none. Hence, intrusion into some low cost IoT devices is not a hard task to many attackers. However, the attackers may have little interest in getting the information of the IoT devices, because the information from a low cost IoT device is usually of limited value. The attackers may also have little interest to send forged messages, because such attacks usually need to be done individually, i.e., IoT devices from different applications have different data format, hence the chances of large-scale data forgery attack is small.

However, attackers may have interest in controlling a large number of IoT devices (including other network devices) to form a botnet, and lunch DDoS attacks on other network service platforms. Such an attack happened in October 2016 when DYN (a DNS service provider) was down due to a heightened DDoS attack, which caused a number of Internet services out of services, including Twitter, Spotify, Netflix, Airbnb, Github, Reddit, and many others. That DDoS attack was found to involve a large number of IoT devices. With the increasing number of IoT devices, and the relatively low difficulty to intrude into IoT devices, and the low communication effort for each individual devices in a DDoS attack, there is an increasing security threat that IoT devices will contribute heavily to massive DDoS attacks.

Different from traditional security goals where the security protection aims at protecting the devices from network attacks, this security threat is not to attack IoT devices themselves, but to master the IoT devices to form a large-scale botnet, and to lunch massive DDoS attacks on other victims in the Internet.

In order to mitigate the chances for the IoT devices to contribute to DDoS attacks by becoming a node in a large scale botnet, an effective security mechanism is to restrict the addresses (e.g., the IP addresses) of network nodes with whom the IoT devices are allowed to communicate, and this setting should not be allowed to manipulate from normal access over the network. Such settings can be changed via a physical connection or a different wireless network port.

6.3 Some Special Features of IoT Perception Layer Security

Security threats come from possible attacks. The purpose of some attacks, or the mechanism of attacks, or the features of security protection techniques, have significant differences from those in traditional information systems and networks. Some of such special security features are described as follows.

Being lightweight. As described earlier, a large number of IoT devices have constrained resources, and their capability to implement security functions is limited. Hence, cryptographic functions having the feature of being lightweight are preferable. Fortunately, quite a few lightweight cryptographic encryption algorithms have been designed and some of them have been practically used.

However, the lightweight feature of public key encryptions and digital signatures seem to have a substantial technical challenge. It is hence proposed that imbalanced public key cryptography is perhaps a new research direction, which means that, for public key algorithms, the design mechanism should seek either the encryption or the decryption to be lightweight, instead of both. For digital signatures, the design mechanism should seek either the signing algorithm (which is preferable) or the verification algorithm to be lightweight instead of both. The idea of designing imbalanced public key cryptosystems is to reduce the computing cost of one algorithm (e.g., the encryption) by sacrificing that of another algorithm (e.g., the decryption).

The lightweight feature is also required in security protocols, e.g., identity authentication protocols. Traditional challenge-response authentication protocols are not lightweight, because the handshake process requires 3-way communication, which requires large amount of energy consumption. Lightweight security protocols for resource-constrained IoT devices have practical demand and need further study.

Hiding to reduce network exposure. Traditional information systems are suggested to have intrusion detection service, so as to reduce the chances of network intrusion. However, many IoT devices do not have the capability to detect intrusion. Noticing that most of such resource-constrained IoT devices do not have the requirement of being connected to the network all the time, and recall that the first step of network attack is to find the potential devices connected to the network, hence an effective and practical mechanism for such IoT devices to reduce the chances of encountering network attacks is to close the network connection when there is no need to send or receive data, hence to reduce the chances of network exposure and the chances of being a target of attack. This simple mechanism can effectively reduce the chances of encountering network attacks.

Dormancy against DoS attacks. DoS attacks on IoT devices are not just to temporarily disable their normal capability to respond, because normal DoS attacks on IoT devices during the time when the devices are not supposed to work make little sense. The purpose of DoS attacks on IoT devices are to make the devices die, energy exhaustion attack is such an attack.

The dormancy mechanism of IoT devices can effectively reduce the energy consumption rate under DoS attacks, because once a connection attempt is verified as invalid, then the device goes into sleep till the next time to wake up. On the other hand, dormancy is also a specific way to hide from network exposure, reducing the chances of being found and becoming a target of DoS attack.

Protection against becoming accomplices in DDoS attacks. While traditional security measures usually consider to protect the IoT devices from any loss or damage causes by network attacks, the protection against becoming accomplices in DDoS attacks is a new security measure to be considered for IoT devices. This is because that damaging IoT devices does not benefit the attackers, while the huge amount of IoT devices can be used to form botnets to lunch DDoS attacks, from which the attackers could possibly extort the victims or potential victims for some ransom.

It is noted that, if some IoT devices are involved in a DDoS attack, then the manufacturers or the operators of those devices have some responsibilities about the loss caused by the DDoS attack and the protection of their IoT devices.

Protection against OT attacks. Operational security is a special security measure in IoT systems, particularly industrial IoT systems. Attackers access to a master computer, and make malicious control over the endpoint IoT devices to cause physical damage. During the attack, the master computer behaves like normal. This is the typical behavior of OT attacks. More description about OT attacks can be found in Chap. 11.

In order to thwart OT attacks, intrusion tolerance mechanism is the most effective approach. For IoT devices, the intrusion tolerance techniques are different from those for information systems, particularly the information systems with sufficient computation capability and complicated configurations. Some intrusion tolerance methods are discussed in Chap. 11.

6.4 Summary

IoT perception layer is where the cyberspace is connected to the physical world, hence the perception layer security is the core component of IoT security.

IoT perception layer consists of IoT devices and some networks for short distance transmission, typically some wireless networks such as Zigbee and WiFi. Security functionalities are realized in IoT devices, however, some IoT devices are resource-constrained compared with normal information systems such as personal computers. The resource-constraint feature of some IoT devices leads to the requirement of lightweight security techniques.

Some characteristics of IoT perception layer security are described. These characteristics can be considered as the differences of IoT systems from traditional information systems, and can help to understand why security techniques for IoT applications (mostly for IoT perception layer) are different.

In the introduction of lightweight security techniques, it is pointed out that public key cryptosystems should seek the imbalanced design. The idea is to make one security function to be lightweight by sacrificing another security function, because to make both of the security functions to be lightweight has a much higher technical challenge. The design of imbalanced public key cryptosystems (or partial lightweight public key cryptosystems) are significant and have important applications in IoT systems, yet it is a problem to be studied.

Finally, it is reminded that, security protection for IoT perception layer (IoT devices in particular) is not to avoid the attacking attempts from attackers, but to increase the hardness of successful intrusion, and to eliminate or mitigate the loss caused by successful attacks.

References

1. A. Bogdanov, L.R. Knudsen, G. Leander G, et al., PRESENT: an ultra-lightweight block cipher, in *Proceedings of 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHESS 2007)*, Lecture Notes in Computer Science, vol. 4727 (Springer, Berlin, 2007), pp. 450–466
2. J.R. Douceur, The sybil attack, in *International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, Lecture Notes in Computer Science, vol. 2429 (Springer, Berlin, 2002), pp. 251–260. https://doi.org/10.1007/3-540-45748-8_24
3. X. Fu, On traffic analysis attacks and countermeasures. Doctoral dissertation, Texas A&M University (2005). http://hdl.handle.net/1969.1/4968
4. A. Shah, M. Engineer, A Survey of lightweight cryptographic algorithms for IoT-based applications, in *Smart Innovations in Communication and Computational Sciences—Proceedings of ICSICCS-2018* (Springer, Berlin, 2018), pp. 283-293. https://doi.org/10.1007/978-981-13-2414-7_27
5. W. Wu, L. Zhang, LBlock: a lightweight block cipher, in *Proceedings of 9th International Conference on Applied Cryptography and Network Security*. Lecture Notes in Computer Science, vol. 6715 (Springer, Berlin, 2011), pp. 327–344
6. W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, I. Verbauwhede, RECTANGLE: a bit-slice ultra-lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. **58**(12), 1–15 (2015)

Chapter 7
IoT Network Layer Security

In the three-layer IoT architecture, the network layer means wide area networks (WANs). This chapter describes some common security techniques of IoT network layer, including those based on TCP/IP network model and those for mobile communication networks. Most security protocols in TCP/IP networks and those in mobile communication networks are industry standards. This chapter briefly introduces some typical network security protocols, including IPSec, SSL/TLS, and the authentication and key agreement (AKA) processes in mobile communication networks.

7.1 Introduction

Although different networks are used in IoT applications, even in the perception layer of IoT, wireless communications are commonly used. For example, the data from an endpoint sensor node to the gate node is often transmitted over wireless networks. However, the IoT network layer is meant to be the wide area networks (WAN), including wired and wireless WANs. The components of IoT network layer is shown in Fig. 7.1.

"The State of the WAN Report 2019" sponsored by Cradlepoint and conducted by IDG surveyed 505 mid-size and larger enterprises with 500 to 10,000 employees, to determine the changing role of LTE and anticipated impact of 5G in the enterprise, as organizations adjust their WAN strategies to deal with the implications of cloud, mobile and Internet of Things (IoT). The research revealed that the LTE usage is increasing and will overtake the broadband, and the impact of 5G networks is even big. Hence, wireless technology will be a core technology in the IoT applications, particularly with the fast development of 5G technology, which will provide a good support to the development of Industrial Internet of Things (IIoT).

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_7

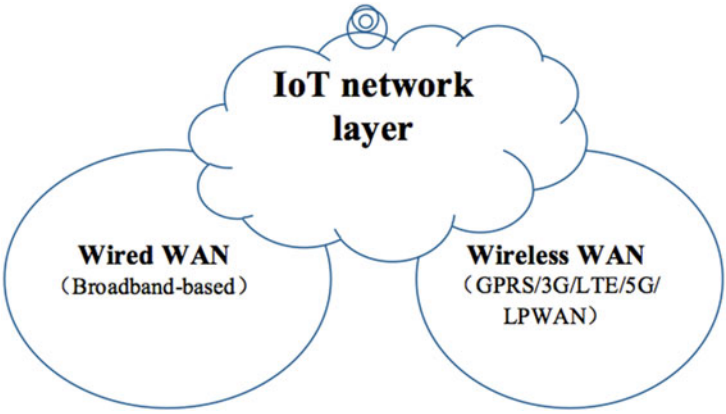

Fig. 7.1 The components of IoT network layer

In recent years, a new type of Wide Area Networks based on narrow band are developed, called *Low Power Wide Area Network (LPWAN)*. LPWAN is a wireless wide area network technology that is targeted to interconnect low-bandwidth, battery-powered devices with low data rates and long range transmission. LPWANs operate at a lower cost with greater power efficiency than traditional mobile networks. They are also able to support a larger number of connected devices.

LPWANs can accommodate packet sizes from 10 to 1000 bytes at uplink speeds up to 200 Kbps. LPWAN's long range is expected to vary from 2 km to 1000 km, depending on the technology. Some may even exceed the range of 1000 km.

Specific LPWANs include NB-IOT, Lora, and Sigfox, they have different architectures, communication protocols, and security techniques. Most of such LPWANs have a star topology, where each endpoint node connects directly to common a central access point (i.e., the gate node).

LPWANs are designed to fit IoT applications, because the characteristics of IoT data are that, many IoT devices generate small size data, low data rate, and large tolerance with time delay. There are also some IoT devices that can stream a significant amount of data, such devices are treated as traditional information systems.

Since the IoT network layer includes different kinds of networks, security issues and techniques of those networks are considered to be in the IoT network layer. This chapter introduces some basic network security protocols, security techniques in mobile communication networks, and security techniques in LPWANs.

7.2 Security Threats in IoT Network Layer

IoT network layer includes wired networks and wireless networks, hence the security threats also come from those networks. Security threats in traditional networks (wired or wireless) remain in IoT network layer, they include the following:

1. Traditional network attacks based on network protocols, such as DNS spoofing attack, ARP spoofing and ICMP spoofing attacks.
2. Network attacks based on data, such as eavesdropping, tempering, and identity spoofing.
3. Network attacks based on services, such as DoS attacks and DDoS attacks. The energy exhaustion attack is a new type of DoS kind of attack on IoT devices with constrained resources.

Countermeasures against network attacks based on network protocols are mainly on the implementations of those network protocols, some associated policies and strategies. Countermeasures against DoS and DDoS attacks are mainly about IP packet tracing and distributed services (e.g., content delivery network). Since the role of IoT network layer is to transmit data from the perception layer to the processing layer (e.g., a data processing center), we only consider the security countermeasures against network attacks based on data.

7.3 Network Security Protocols

As an important component in IoT network layer, Wide Area Networks (WANs) have wide bandwidth and are usually shared by different user groups, resulting in the networks to be publicly accessible. The public access introduces security threats, including eavesdropping, message tempering, and identity fraud. The corresponding security techniques are typically encryption, message authentication code, and identity authentication protocols [5].

In the network environment, security services are implemented by security protocols. Traditional security protocols for IP-based networks include IPSec, a security protocol suite based on IP layer of TCP/IP model, SSL/TLS, a security protocol suite build on top of TCP layer of TCP/IP model, and many application layer security protocols. Some typical security protocols are briefly introduced here, simply to show how they work, i.e., how the provision of security services (e.g., authentication, key management, confidentiality, integrity) are implemented.

It is noted that wired networks and wireless networks all have their advantages and disadvantages. Wired networks are inconvenient to use, and difficult to eavesdrop, because accessing the communication signals is relatively difficult, particularly when the communication media is optical fiber. Wireless networks are convenient to use, and easy to eavesdrop, because wireless signals transmit to everywhere.

7.3.1 Network Architecture Models

The Internet is a super large computer network, and is supported by a large number of network protocols, called *Internet Protocol Suite*. The protocol suite is a combination of protocols which encompasses a number of different protocols for different purposes. Because the two major protocols in the suite are TCP (Transmission Control Protocol) and IP (Internet Protocol), the internet protocol suite is commonly referred as TCP/IP protocol suite [1]. This protocol suite has its own reference model. This model has four logical layers in contrast with the OSI model which has seven layers.

The OSI model is only a reference model, no protocol is defined. Hence, the TCP/IP still dominates the practical implementations. In fact, these two models do not have substantial difference. By combining some of the layers as a single layer, the 7-layer OSI reference model can correspond to the 4-layer TCP/IP. The relations between these two reference models and their logical correspondence can be depicted in Fig. 7.2.

In the TCP/IP model, there are different protocols defined in different layers. For example, the Network interface layer has *Address Resolution Protocol (ARP)* and *Reverse Address Resolution Protocol (RARP)*; the Network layer has *Internet*

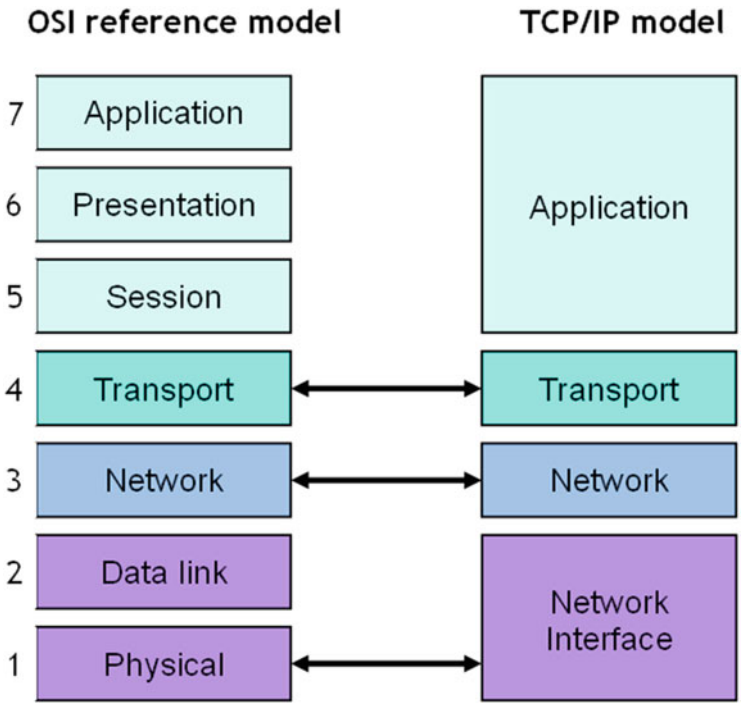

Fig. 7.2 Comparative depiction of OSI and TCP/IP reference models

Protocol (IP), the Transport layer has *Transmission Control Protocol (TCP)* and *User Datagram Protocol (UDP)*, and the Application layer has quite a few protocols, including *Domain Name System (DNS), File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP)*, and *Telnet*, they refer to different applications.

The TCP/IP protocol suite was designed for the purpose of message transmission, while security problems were not considered. However, with the development of the Internet applications, security threats became obvious, and security protocols are added into the network protocol suite.

Security protocols can be added into different layers. On top of the Network layer, *Internet Protocol Security (IPSec)* was proposed. IPSec can be used to establish a virtual private network (VPN). Based on TCP layer, *Secure Sockets Layer (SSL)* was proposed, which is a security layer lies between the Transport layer and the Application layer, and re-named as *Transport Layer Security (TLS)* since version 3.1. In the Application layer, a number of security protocols are proposed, some of them are based on SSL, such as *HyperText Transfer Protocol over SSL (HTTPS)*, while some others are independent security protocols, such as *Pretty Good Privacy (PGP)*.

7.3.2 Internet Protocol Security (IPSec)

The Internet Protocol Security (IPSec) was designed by the Internet Engineering Task Force (IETF), targeting at providing security protection for the Network layer via end-to-end encryption and authentication. IPSec is actually a collection of sub-protocols, which is composed of two parts: one part is to exchange keys for data encryption and data source authentication, where the sub-protocol is called *Internet Key Exchange (IKE)*. The other part is to do the encryption and authentication, including sub-protocols *Encapsulating Security Payload (ESP)* and *Authentication Header (AH)*. The AH is to authenticate the sender and any possible illegal modification of data during data transmission. The ESP is to perform authentication of the sender and encrypt data. The details of the sub-protocols are specified in RFC documents, as briefly described in Table 7.1.

There are two modes of IPsec in its ESP, they are as follows:

Tunnel Mode: This mode takes the whole IP packet to form secure communication between two places, providing point-to-point security service. A specific

Table 7.1 The RFC documents for IPSec

RFC number	Name of document
4301	Security architecture for the internet protocol
4302	IP authentication header
4303	IP encapsulating security payload (ESP)
4306	Internet key exchange (IKEv2) protocol

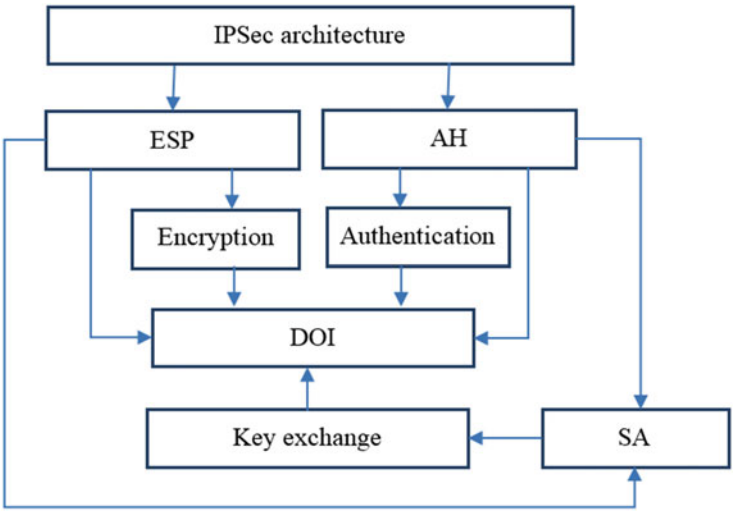

Fig. 7.3 The architecture of IPSec

application of this mode is to setup a security tunnel between two network gates (e.g., firewalls).

Transport Mode: This mode only encapsulates the IP payload (not the entire IP packet as in tunnel mode) to ensure a secure channel of communication, providing end-to-end security service.

When using the IPSec, there is a *Security Association (SA)* specifying the security parameters to be agreed between communication parties. There is also a *Domain of Interpretation (DOI)*, which is used to group related protocols using the *Internet Security Association Key Management Protocol (ISAKMP)* to negotiate security associations. The sub-protocols and their relations can be depicted in Fig. 7.3. Details of the sub-protocols can be found in most network security books.

7.3.3 Secure Socket Layer (SSL)

Secure Socket Layer (SSL) is a standard network security protocol, consisting of a few sub-protocols, lying in between the Transport layer and the Application layer of the TCP/IP model (see Fig. 7.4).

SSL was designed by Netscape for establishing an encrypted link between a server and a client, typically a web server and a browser. SSL uses public key infrastructure to establish security services based on TCP connection, and provides mutual authentication between the server and the client.

The latest version of SSL is version 3.0. In 1999, the version 3.1 of SSL was named as *Transport Layer Security (TLS)* version 1.0, and became a standard of The

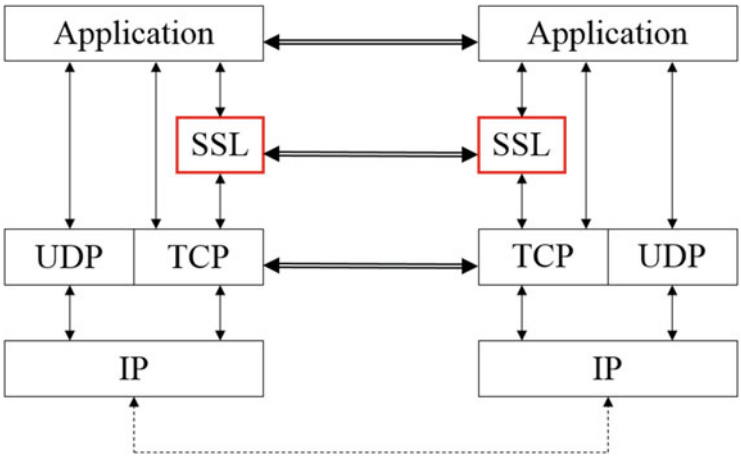

Fig. 7.4 The logical position of SSL

Internet Engineering Task Force (IETF) and specified in the RFC-2246 document. TLS has a number of improvements over SSL, supporting newer and more secure algorithms, including message authentication, key generation, and the supported cipher suites.

When SSL is used by web services, the basic process of SSL protocol is as follows:

1. Client connects to web server by sending a "HELLO" message, and gives a list of available ciphers.
2. Server picks the strongest cipher that both it and the client support, and sends back its public key certificate issued by a trusted Certificate Authority (CA).
3. The client verifies the validity of the certificate using the CA's public key, and obtains the server's public key. In practice, clients are assigned a collection of CAs public keys locally, so this can be done without having to contact the CA.
4. The client generates a random number, encrypts it with the server's public key, and sends the ciphertext to the server. The server decrypts the ciphertext (using its private key) to get the random number.
5. Server and client use this random number to generate session keys to establish a secure communication channel.

The SSL protocol stack is composed of some sub-protocols, they include *SSL Handshake Protocol, SSL Change Cipher Spec Protocol, SSL Alert Protocol*, and *SSL Record Protocol*. The SSL architecture can be shown in Fig. 7.5.

The Handshake protocol and the Record protocol have to be involved in every execution of SSL protocol, while the Change Cipher Spec Protocol is used when there is a need to change cipher specification such as cipher name, updated key, or encryption mode for block ciphers. The Alert protocol is used only when something goes wrong and an alert message needs to be sent by the SSL protocol. The process

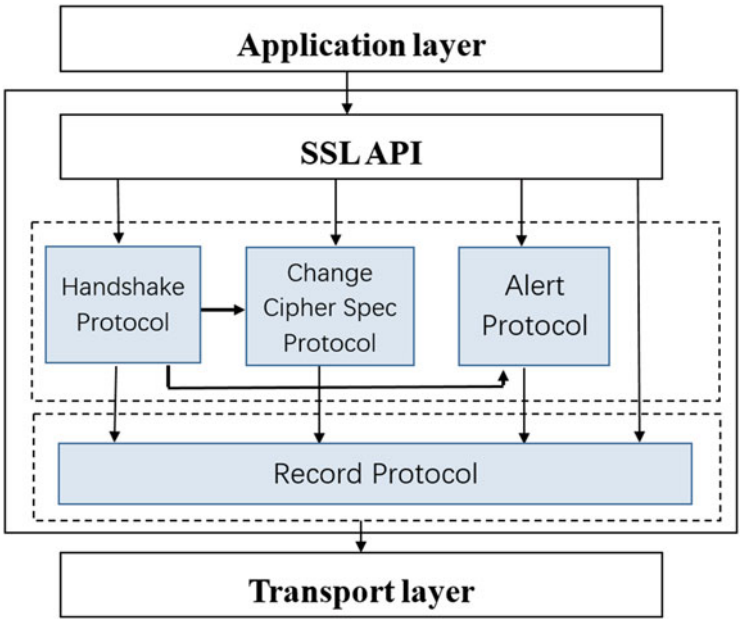

Fig. 7.5 The architecture of SSL

of SSL connection can be depicted in Fig. 7.6, where the message marked with an asterisk (*) means optional, and the *Change Cipher Spec protocol* (same as *Alert protocol*) is put into square brackets, meaning occasional.

Although the initial design of SSL was to provide security services for web connections, it can be used for any application with a client and a server, hence can be used for different applications. SSL is designed to support multiple connections between a specific client and a specific server within a same session. This means that in every session of SSL protocol, the Handshake protocol and the Record protocol have to be executed, and random number for key generation are shared. Then when the SSL session is not terminated, new application connections can use the same SSL security parameters to setup new secure channels. This manner of function can be depicted in Fig. 7.7.

Although the SSL/TLS protocol does not seem to have severe security vulnerability, its implementation may introduce vulnerability due to implementation flaw. In 2014, the popular OpenSSL cryptographic software library was found to have a serious vulnerability, which was known as *Heartbleed bug*. The SSL Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt data, including the names and passwords of the users and the actual communication content. This event alerts that a secure protocol may not guarantee security service in practical implementations.

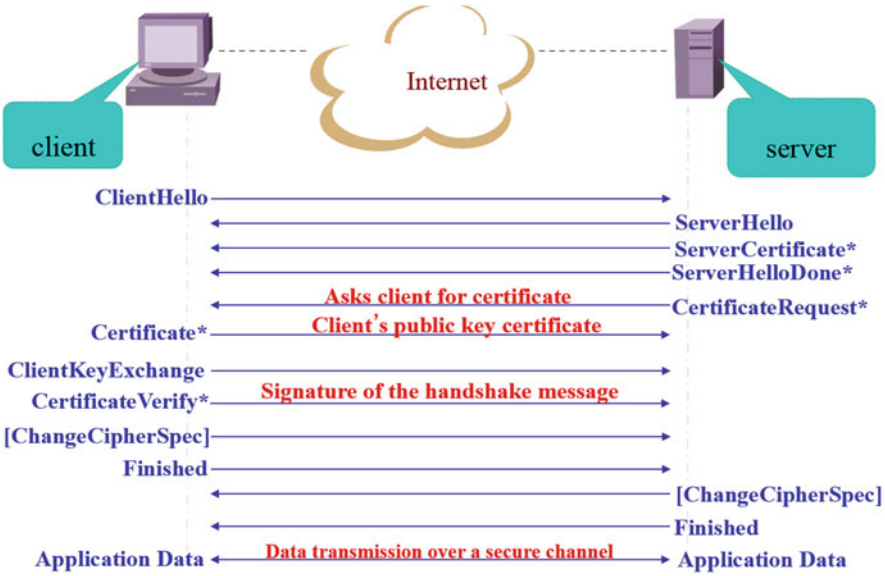

Fig. 7.6 The process of SSL connection

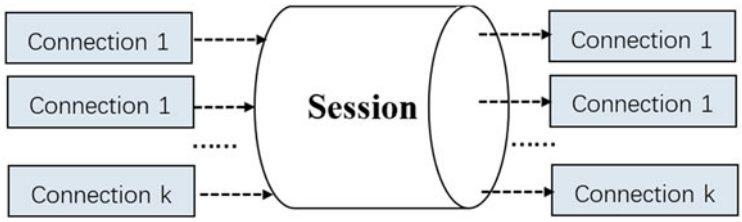

Fig. 7.7 One SSL session supports multiple connections

7.4 Security Techniques in Mobile Communication Networks

Mobile communication networks have been evolved dramatically from 2G, 3G, LTE (some versions of LTE are also referred to as 4G), and 5G [2]. The first generation of mobile communication networks were based on analogue signals, and had no security. Since the second generation (2G) of mobile communication networks, the mobile communication networks use digital technology and can provide some security mechanisms. The Global System for Mobile (GSM) as 2G mobile communication networks have some security vulnerabilities (e.g., no server authentication). The 3rd generation (3G) of mobile communication networks improved the security vulnerabilities, including enhanced security algorithms and message integrity services. The Long Term Evolution (LTE) networks further enhance the security services than those in 3G, while providing higher speed data

transmission. The 5G networks have wide bandwidth and low communication delay, where the detailed security techniques are not quite clear due to limited documents available.

7.4.1 Security Techniques in 2G Mobile Communication Networks

In 2G mobile communication networks, there are three entities: the user equipment (UE), which is typically identified by a subscriber identity module (SIM, commonly known as SIM card), in combination with a cell phone. A SIM card provides a temper-proof environment, holds some secret information and executes security algorithms. The mobile network can be represented by two entities, the home location register (HLR) and a visitor location register (VLR). The HLR is where the user SIM card is issued, and the HLR is where the roaming user gets mobile communication services. Practically, an HLR manages a number of neighboring base stations.

The HLR works as an authentication center by managing the information (including the secrets) of the SIM cards. Each SIM card has a unique ID, known as international mobile subscriber identity (IMSI), and a secret key k embedded in the SIM card is shared with the HLR. The IMSI and the secret key k will be used for the subscriber to authenticate itself to the network.

When a UE tries to connect to the network, it first tries to reach a nearby base station, sends the ID of the UE to the base station, which collects the information and sends it to VLR, who then communicates with the HLR to authenticate the UE. When the authentication is done, a session key is agreed between the UE and the network, the VLR in particular, because the VLR is responsible for the wireless connection with the UE. Hence, the authentication process is also called *authentication and key agreement (AKA)*. The process of AKA can be depicted in Fig. 7.8.

From Fig. 7.8 it can be seen that, when a UE tries to connect to the mobile networks, it first broadcasts its identity IMSI, which is expected to be received by the nearest VLR. The VLR is able to find which HLR the IMSI belongs to, and sends the IMSI to the corresponding HLR. The HLR finds the user key k, creates a number of authentication vectors, in the form of triples $(RAND, XRES, Kc)$, and sends the authentication vectors securely to the VLR. It is assumed that the communication between the VLR and the HLR is secure. Each authentication vector is composed of the following: a 128-bit random number $RAND$, a 32-bit expected response $XRES = A3(k, RAND)$, and a 64-bit data encryption key $Kc = A8(k, RAND)$, where $A3$ and $A8$ are two standard encryption algorithms in the GSM system. When the VLR receives the authentication vectors, it chooses one of the authentication vectors, sends the RAND in the authentication vector to the UE. When the UE receives the RAND, its SIM card has the user key k and the encryption algorithms

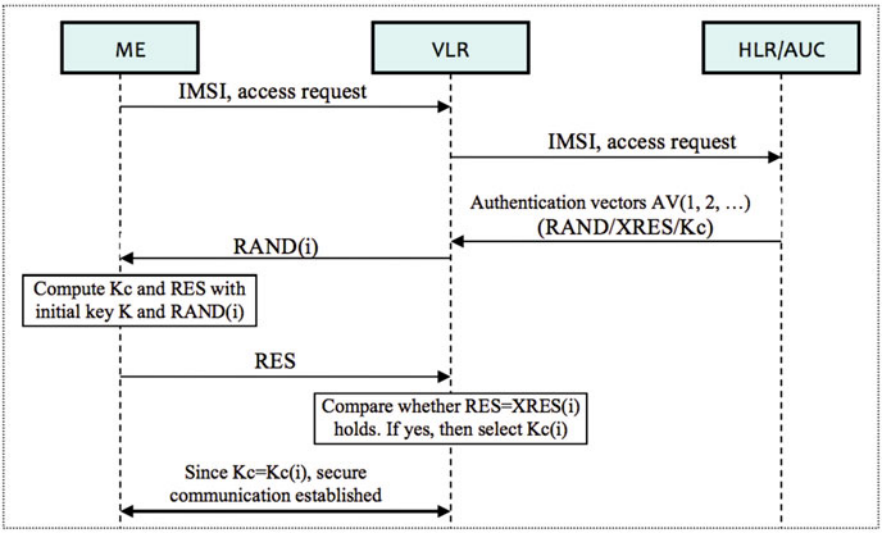

Fig. 7.8 The authentication and key agreement process of 2G mobile communication networks

$A3$ and $A8$, hence can compute $RES = A3(k, RAND)$. UE sends RES to the VLR. The VLR then compares the RES received from the UE and the $XRES$ from the authentication vector. If the equality RES=XRES holds, then the UE is authorized, and a temporary session key Kc is shared between the UE and the VLR, with an encryption algorithm $A5$, a secure communication channel can be established between the UE and the VLR.

When the VLR succeeds the process of UE authentication, it generates a temporary mobile subscriber identity (TMSI), and sends it to UE. The next time when the UE is required to make a new connection, it first uses the TMSI as its identity, and the authentication process is much simpler. However, if the connection request is with a new VLR, then the whole AKA process has to be repeated.

The old version of the encryption algorithms $A8$ and $A5$ were broken. Breaking the encryption algorithm $A8$ means that the SIM cards can be cloned, and breaking the encryption algorithm $A5$ means that the communication between the UE and the VLR can be eavesdropped and decrypted. Later on, the algorithms $A8$ and $A5$ were upgraded. However, the security mechanism of 2G networks has the vulnerability that the authentication on the network is missing, which is the reason for fake base stations to cheat mobile users. Fortunately, the 3G mobile communication networks solved the security vulnerabilities found in 2G networks.

7.4.2 Security Techniques in 3G Mobile Communication Networks

In the third generation (3G) of mobile communications networks, there is a big change on the security mechanism. 3G network enables mutual authentication between UE and the serving network, uses new cryptographic algorithms and new functionalities.

The topology of 3G networks is like that of 2G networks, but 3G networks use some new terms. The subscriber device is called *Universal Subscriber Identity Module (USIM)*, the HLR is also called authentication center, and the VLR is also called *Serving GPRS Support Node (SGSN)*. The authentication and key agreement process is an *Extensible Authentication Protocol (EAP)*, which can be depicted by Fig. 7.9.

In Fig. 7.9, when a UE requests for network access, it sends its identity IMSI to the network, which is forwarded to the HLR by SGSN. The HLR generates a number of authentication vectors, each authentication vector is a 5-tuple (RAND, XRES, CK, IK, AUTH), where RAND is a random number, XRES is expected response from the UE, CK and IK are encryption key and integrity protection key (for MAC algorithm) respectively, and AUTH is an authenticator for the UE to authenticate the network. This means that mutual authentication is enabled, and the presence of IK means that message integrity protection is provided.

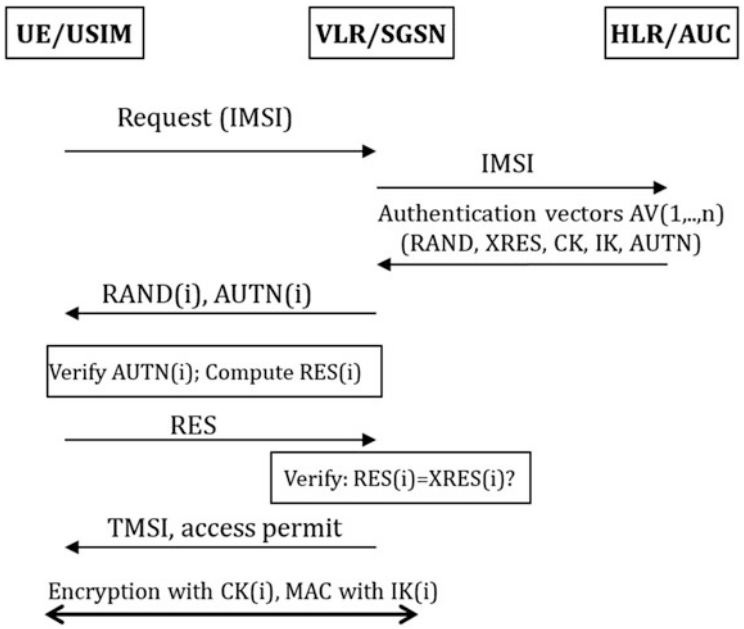

Fig. 7.9 The authentication and key agreement process of 3G mobile communication networks

The problems with 3G networks are that, there are a number of standards, namely WCDMA, TD-SCDMA, and CDMA2000. Different standard uses different technologies, and there are different types of mobile equipment, i.e., a mobile station using WCDMA network cannot be used to access the TD-SCDMA network or CDMA2000 network, even if a new USIM card is used. This leads to the need for a better mechanism, hence LTE was evolved.

7.4.3 Security Techniques in LTE Mobile Communication Networks

Given the problems existed in 3G networks, the long term evolution (LTE) networks were designed. Compared with the 3G networks, the LTE networks are around five times faster, and have a number of other advantages.

LTE uses the technologies of orthogonal frequency division multiplexing (OFDM) and multiple input multiple output (MIMO), providing about 326 Mb/s upstream and 86 Mb/s downstream data rate with 20 MHz signal frequency.

The term LTE is not connected to which generation of mobile networks, and people are interested to know whether LTE is 4G. In March 2008, the radio sector of the International Telecommunication Union (ITU-R) set standards for 4G connectivity, requiring all services described as 4G to adhere to a set of speed and connection standards.

When the LTE networks were developed to an advanced stage, called *Long-term Evolution Advanced (LTE-A)*, they meet almost all the requirements of 4G, and people tend to call LTE-A as 4G networks.

LTE networks have some core entities, namely the user equipment (UE), evolved node base (eNB), mobility management entity (MME), and home subscriber server (HSS). The UE is as in 3G networks; eNB manages wireless resources; MME has a number of functionalities, like the VLR in 2G/3G networks; and HSS is responsible for user authentication like the case in HLR in 2G/3G networks.

The security mechanism of LTE is a bit more complicated than that of 3G. There are two layers of security protection, one security layer is from UE to eNB, and the other security layer is from eNB to MME. The detailed processes are beyond the scope of this book.

7.5 Security Techniques in LPWAN

Low power wide area network (LPWAN) is not a specific network, but a general term for a collection of networks with similar services, i.e., they target at serving IoT devices with constrained resources, hence the LPWAN provides long distance communication with low power consumption. Inevitably, this would cause some

transmission delay, which is tolerable to most of the resource-constrained IoT devices. The feature of low power consumption means that the size of transmitted data is small, this is also the practical situation for the resource-constrained IoT devices. So, the LPWAN well meets many IoT applications. A number of LPWANs have been designed, such as NB-IOT (previously LTE-M), LoRa, SigFox, NWave, OnRamp, Platanus, Telensa, Weightless, Amber Wireless. Some of them have been well commercialized, such as NB-IOT and LoRa.

7.5.1 Security Techniques in NB-IOT

NB-IOT is based on current LTE network infrastructures, it is a standard of Third Generation Partnership Project (3GPP), the organization behind the standardization of mobile communication networks. NB-IOT is designed to address the needs of very low data rate devices that are often powered by batteries. Since the construction of NB-IOT is based on the available infrastructure of mobile communication networks, the construction can be made quite repaid, and can have a seamless connection with mobile communications.

Since the NB-IoT networks are constructed based on the current mobile communication networks, which are currently dominated by LTE networks, the identity authentication and data security protection techniques in NB-IOT are tailored from those of the base networks. However, NB-IOT uses an independent 180KHz bandwidth beyond the existing mobile communication networks, hence do not occupy the normal bandwidth of LTE for normal voice and data transmission.

The core network components of NB-IOT include mobility management entity (MME), which is much the same as that in LTE, a serving gateway (S-GW), and a PDN gateway (P-GW). NB-IOT construction adopts the technology of virtualization, making the Evolved Packet Core (EPC) to be virtual, including MME, S-GW and P-GW.

As an endpoint equipment in mobile communication networks, NB-IOT endpoint devices are supposed to have a SIM/USIM card, so that the security functionalities can be realized. However, with the size of IoT devices becoming smaller and smaller, although the size of SIM/USIM cards also become smaller than before, it is still a big challenge for small IoT devices to use SIM cards in normal size. Considering that the chances for an IoT device to change the SIM/USIM cards are small, it would be a good idea to integrate the SIM/USIM with the devices. However, there is a wide range of manufactures making IoT devices, while the manufacture for SIM/USIM should be supervised with strict security requirement, this makes the integration difficult. As a tradeoff, the use of electronic SIM (called e-SIM) or soft SIM (called SoftSIM) can be a good solution. The technology of eSIM was proposed by Global System for Mobile Communications Assembly (GSMA), and the implementation techniques and commercial target have much difference with those in SIM/USIM.

In the NB-IOT core network, MME is responsible for UE authentication, can establish and terminate a secure channel between the UE (essentially the SIM) and the network, and select proper S-GW and P-GW according to the configuration of UE. Although the UE in mobile networks is supposed to be mobile, the NB-IOT, however, is best suited for primarily static devices, like meters and sensors in a fixed location, rather than roaming devices.

NB-IOT supports wireless update of software and firmware, and the process of update should require digital signature for authentication and public key encryption for convenient data encryption. When a UE receives the data package for software or firmware update, it verifies the correctness of the digital signature, and performs decryption to get the correct data. This security mechanism ensures that the update is authentic, so that malicious software update for the UE can be avoided.

7.5.2 Security Techniques in LoRa

LoRa was developed by a company named Semtech, it is very different from NB-IOT. LoRa makes use of unrestricted frequency bands below 1 GHz, with typical frequencies being 433, 868, or 915 MHz, enabling long distance and low power data transmission [4]. The LoRa chips were announced in August, 2013. Since then, a large number of companies formed the LoRa Alliance.

The differences between LoRa and NB-IOT are obvious. The comparison of some behavior of Lora and those of NB-IOT are given in Table 7.2.

Due to the different operation of LoRa and NB-IOT, users make their payment differently. For the NB-IOT, the payment is like that for mobile phone users, which can be made on data package, monthly payment with or without data limitation, while the LoRa users often choose to buy-out the LoRa devices.

Considering that the resources of IoT devices vary largely, it is not possible to accommodate all the devices properly. LoRa can be properly configured to provide services for three different classes of devices, namely Class A, Class B, and Class C, they are used for different functionalities.

Table 7.2 Behavior comparison between LoRa and NB-IOT

Behavior	NB-IOT	LoRa
Technology	Cellular	Multi-hop mesh networks
Network topology	Star	Star
Spectrum	Licensed frequency bands	Unlicensed Spectrum
Transmission distance	Long distance	Long distance (1-20KM)
Data rate	<100 Kb/s	0.3–50 Kb/s
No. connections/cell	200K	200–300K
Data security	Support	128-bit AES
Battery life	3–10 years	3–10 years

Class A: This class of devices support two-way communication. Every time when data uploading is completed, there are two small windows listening to possible downlink messages. If none message is received, then the devices turn into dormancy until next wake up time. This class of devices are the most energy-saving.

Class B: This class of devices can be waked up at any time by a Beacon signal sent from the network gateway. However, the endpoint devices may not be listening to the network all the time. This class of devices consume more energy than the Class A ones due to the network listening functionality.

Class C: This class of devices are awake all the time, either they are in the process of sending data, or in the process of listening to the network. This class of devices consume the most energy.

In order to protect data confidentiality during data transmission, LoRa uses the counter mode of AES128 to encrypt data, which works like a stream cipher, making the encryption to be XOR of the plaintext data and the key-stream generated by the counter mode of AES128. Note that the use of counter mode need a counter for one-way communication, hence LoRa needs two counters to enable two-way secure communication. The counter values are recorded as FCntUp and FCntDown for up-stream and down-stream data transmission respectively, they are maintained and updated by endpoint devices and the data server respectively. The process of key-stream for data encryption can be depicted as below:

```
For i = 1, \ldots , k= ceil(len(FRMPayload)/16), compute
  Ai = (0x01||(0x00*4)||Dir||DevAddr||FCntUp/FCntDown||0x00||i)
  Si = AES128_encrypt(K, Ai),
S = S1||S2||\ldots ||Sk
```

It is noted that the process of key-stream generation uses either FCntUp or FCntDown, this ensures that the key-stream will never repeat within the lifetime of the endpoint devices.

7.6 Summary

The network layer of IoT is the tunnel for data transmission between the perception layer and the processing/application layer. It makes use of different technologies for wide area networks for long distance data transmission, making the IoT a real Internet of Things.

This chapter briefly describes some typical network security protocols, the security mechanism of mobile communications, and superficial description about some security techniques in LPWAN.

Since the IoT network layer can be treated as a collection of traditional WANs, security issues can refer to those in traditional WANs. An exception is the new LPWAN technology, which is specially designed for IoT applications, targeting at providing long distance communication services with low power consumption.

Security mechanisms of different LPWAN technologies are similar, while the technical implementations can be quite different.

It is anticipated that the commercialization and further optimization of 5G networks will make 5G technology a favorable choice for IoT network layer. However, since we have little knowledge about security mechanism and detailed techniques in 5G mobile communication networks [3], 5G network security is an area for further study.

References

1. C.M. Kozierok, *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference* (No Starch Press, San Francisco, 2005). ISBN-13: 978-1593270476
2. A. Kukushkin, *Introduction to Mobile Network Engineering: GSM, 3G-WCDMA, LTE and the Road to 5G* (Wiley, Hoboken, 2018). ISBN-13 : 978-1119484172
3. J.T.J. Penttinen, *5G Explained: Security and Deployment of Advanced Mobile Communications* (Wiley, Hoboken, 2019). ISBN-13: 978-1119275688
4. T. Petri, M. Goessens, L. Nuaymi, et al. Measurements, performance and analysis of LoRa FABIAN, a real-world implementation of LPWAN, in *Proceedings of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* (IEEE, Piscataway, 2016). https://doi.org/10.1109/PIMRC.2016.7794569
5. W. Stallings, *Cryptography and Network Security – Principles and Practice*, 7th edn. (Pearson, London, 2016). ISBN-13: 978-0134444284
6. Y.P.E. Wang, X. Lin, A. Adhikary, et al., A primer on 3GPP narrowband Internet of Things (NB-IoT). IEEE Commun. Mag. **55**(3), 117–123 (2017). https://doi.org/10.1109/MCOM.2017.1600510CM

Chapter 8
IoT Processing Layer Security

This chapter introduces security threats, security requirements and common techniques of security protection in IoT processing layer. A typical IoT processing layer is cloud computing, hence the security of cloud computing can represent the IoT processing layer security. Cloud computing security includes cloud computing platform security, security of cloud computing services, and data security in cloud computing environment. As in other information systems, access control techniques are described as part of IoT processing layer security. Cloud computing platform also uses the technique of virtual computing, and the security mechanism of virtual computing is introduced.

8.1 Introduction

The processing layer of IoT is referred to a platform providing data processing services, it is not restricted to the scale of the processing platform. The processing platform can be big or small. If the IoT data are processed by a workstation, then the workstation is the processing platform. With the development of cloud computing technology, cloud computing platform can provide a variety of services for IoT data processing. Hence, the processing layer of IoT is mainly about the cloud computing platforms.

Cloud computing technology is designed to handle super large amount of data, called *big data*. There is no formal definition about what big data is, however, the nature of big data is that, big data can hardly be handled using traditional systems, for example, the size of data exceeds the storage capability of any single traditional computer or workstation. Moreover, in cloud computing, it can handle a large number of concurrent processes that would exceed the computing capability of a single traditional computer or workstation. The cloud computing technology is not just about the computing capability increase, it has many changes compared

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_8

with traditional way of data processing. Cloud computing platforms have their own methods of data storage, data backup and recovery, and the methods to provide services.

In a large-scale IoT application system, data play the key role. The data in IoT application system, in an overall scale, have the following characteristics:

Large amount of data. An IoT processing platform (e.g., a cloud computing platform) may serve many IoT applications, even a single IoT application system can have a large amount of data, depending on the scale of the application, the capability of the terminal IoT devices, and the precision of the data. When the data from a large number of IoT applications are put together, it can be as much as terabytes (TB), and may be continuously growing. The smart city is such a platform, it has data from different applications, including smart traffic, smart grid, and electronic government.

A variety of data structure. IoT data are of different structure. Some data have fixed structures, such as temperature, moisture, wind power, and altitude, while others may not have fixed structure, such as image, video, audio, and some ciphertexts. Structured data can be stored in a relational database such as SQL, while unstructured data can be stored in non-relational databases, data lakes, and data warehouses.

Interactions between different data. IoT data may not be independent, different IoT data may be closely related. This means that the data change in one IoT application may affect the data change in another IoT application. Even within a same IoT application, the change of one type of data may affect the change of a different type of data. For example, data change in smart traffic system may affect the schedule of smart transportation, and data change in temperature may affect the operation of ventilation.

Open access: While the role of IoT processing layer is data processing, the processing result is for the eventual users. User accessing to the processing layer hence becomes unavoidable. Since the user access may come from anywhere, this makes the processing layer (e.g., the cloud computing platform) publicly accessible, although the user access is often restricted by identity authentication.

Due to the possible complexity of data in the IoT processing layer, there exist different types of security threats. Hence, sophisticated security techniques are needed to safeguard the proper function of the IoT processing layer.

8.2 Security Threats in IoT Processing Layer

Considering the IoT processing layer as a big data processing platform, there are different types of security threats, including the following typical types:

1. **Forged data source.** In the IoT processing layer, data may come from the Internet, or the mobile communication networks. In either of the cases, the

claimed (or marked with address information) data source may not be the real one. If such data is accepted, it may lead to a data forgery attack. In order to avoid accepting forged data, data source authentication is necessary.

2. **Unauthorized access.** Since the IoT processing layer handles a large amount of data that are used for end users, the processing layer also provides services for the users to access their application data. However, some users may attempt to access other users' application data, including unauthorized reading, writing, modifying the data. Although access control policies are supposed to restrict users to their own rights, security flaws always exist which may be used by malicious uses to gain unauthorized access.

   In order to minimize the effect of unauthorized access, users may choose to encrypt their confidential data, while the platform serving as the process layer should have data backup and data recovery services. In this case, unauthorized access cannot recover the content of the confidential data. When some data is damaged, modified, or illegally created, the data recovery service can recover the data at the status of an earlier time.
3. **Denial of service attack.** In the network environment, an IoT processing layer is exposed to the Internet all the time, hence denial of service (DoS) attacks and distributed denial of service (DDoS) attacks are common security threats. With the development of IoT applications and increased number of IoT devices are connected to the Internet, DDoS attacks from a large-scale botnet can have enormous attack strength.
4. **Internal attack.** Internal attacks are those lunched by some internal users, including super-user accounts. Such attacks are rarely considered, because they can hardly be prevented. However, a proper management policy and technical mechanism can effectively mitigate the effect of internal attacks. Technical mechanism including right separation, i.e., big decision and critical operation cannot be done by a single user, but be accomplished by the operation of a number of users/managers, while management policy includes how to logically separate those managers, so that the chances of their colluding is minimized.

Give a variety of security threats in the IoT processing layer, security techniques are needed to protect the data security and proper operation. Security services needed in the IoT processing layer include network access (e.g., identification and authentication, access control), data storage security (e.g., confidentiality, integrity, backup and recovery), database security (e.g., secure query), system security (e.g., security audit, timely patching, virtual computing management), and security computation (e.g., secure out-source computation service).

8.3 Database for IoT Processing Layer

The IoT processing layer usually needs to process different types of data, and has different types of database.

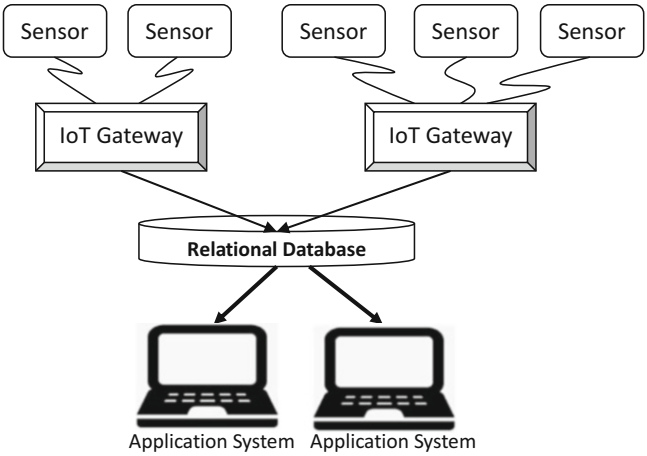

Fig. 8.1 IoT structure with a relational database

The most common database is relational database. Relational database has strict mathematical properties, and has many advantages in database query and other processes, hence, relational database was the most common database in 1970s. As a result, a specific structured query language (SQL) was designed, and was widely used by industry.

Naturally, relational database is also a common database for IoT applications. The structure of relational database-based IoT systems can be depicted in Fig. 8.1.

Relational database has over 50 years of history, both academic and industry have conducted deep study on it, and a number of industrial standards have been made available to safeguard the application of relational database. Among those standards is the Trusted Database Interpretation (TDI), which was developed specifically for database. The TDI standard integrates the Trusted Computer System Evaluation Criteria (TCSEC) into database management.

Non-relational database is designed to accommodate the unstructured data in the Internet era. This is due to that the contents of web pages and multimedia do not fit features of relational database. The techniques and industrial standards of non-relational database are not as mature as those of relational database.

8.4 Access Control Policies Applicable to IoT Processing Layer

Access control is an important security mechanism for computer operating systems, it is also important for cloud computing platforms.

Basically, when considering database access, access control can be considered as a logical policy guiding how a database is accessed. Access control is about how a

subject can do (according to access policy) with respect to an *object*. So the three components defined by access control are as follows:

Subject: A subject is an entity that initiates an action. A typical subject is a user.

Object: An object is a kind of data resource that can be used by a subject. Tables, images, computer files, and database, are all different objects.

Access policy: Access control policy defines rules about how a subject can access to an object. The process of access has to follow the rules defined by the access control policy, otherwise, the action made by the subject will be declined.

There are different policies of access control. Typical access control policies include discretionary access control (DAC), mandatory access control (MAC), and role-based access control (RBAC). Some basic access control policies are as follows:

Discretionary Access Control (DAC): DAC was originally defined by the Trusted Computer System Evaluation Criteria (TCSEC) as "a means of restricting access to objects based on the identity of subjects and/or groups to which they belong". The controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject. DAC has been the most popular access control policy used in computer systems. It was used as early as 1970s in the UNIX operating system. DAC defines a set of rights for each subject-object relation. Typical rights are *read, write, execute*. If a subject A has the *read* right to the object X, then A can read X. If the subject A has the *write* right to the object X, then A can create and/or modify X. If subject A has the *execute* right to the object X, then A can execute X, provided that X is an executable file. A subject may have multiple access rights on an object. If a specific right is not enabled, then the subject cannot access the object to do the corresponding operation. Since the access rights are defined for each subject-object pair, a specific DAC policy can be described by a matrix, called *access control matrix*.

In DAC, a subject can grant its own access rights to other subjects. For example, if user Alice has the right to read the file *Employee*, then she can grant this right to Bob.

A security flow is that, if a user device is affected by a Trojan horse virus, then the Trojan horse can grant the user's rights to anyone who can do whatever the user can. This makes the DAC policy vulnerable to systems with very sensitive data.

Mandatory Access Control (MAC): MAC was designed to overcome the security flaw of DAC policy. Under MAC policy, the management of access control rights belongs to a system administrator, while normal users (subjects) do not have the capability to change the access rights. The MAC policy is suitable to military and government users, where the security requirement is tougher than a system for public users.

MAC policy defines different security attributes, such as top secret (T), secret (S), confidential (C), and unclassified (U). The security attributes satisfy the following order: $T > S > C > U$, i.e., the security level of attribute T is

higher than that of the attributes S, C, and U, while the security level of attribute C is higher than that of attribute U, but lower than that of attributes T and S. Every subject and object is assigned a security attribute. If the level of security attribute of subject A is higher than or equals to that of object X, then A can access X. Otherwise, A cannot access X, even if the owner of X is willing to let A access X.

The typical implementation of MAC policy is Bell-LaPadula (BLP) security model [12] and Biba security model. The BLP model prevents confidential data to be accessed from higher level security attribute to lower level, while the Biba model further prevents data modification from lower level security attribute to higher level.

Role-Based Access Control (RBAC): RBAC combines the features of DAC and MAC. A typical implementation of RBAC policy is RBAC96 designed by Sandhu et al. [7]. In RBAC, permissions are associated with roles, and users are grouped by different roles. Each role group has a delegate user who can authorize another user to become a member of the delegated role group.

RBAC complies with three principles:

Minimum right. The RBAC introduces the concept of session. In a session, a user is granted a role with minimum right to accomplish a designated task. Once the task is accomplished, the session is terminated and the assigned role is ended.

Duty separation. A user cannot have conflicting roles. If a user is assigned two roles with overlap permission, then it may cause a security flaw, i.e., the user is not allowed to do something with one role while he/she is allowed to do so with another role.

Defined right. Apart from the conventional right such as read, write, and execution, RBAC allows other rights to be defined. For example, mortgage upper limit can be defined as a new right for the role of lender.

Compared with DAC and MAC, the policy in RBAC is moderate in the sense that, sometimes it appears to be more like DAC, and sometimes it appears to be more like MAC. In addition, the concept of role can be defined the same way as in real life, which is convenient to use and easy to understand.

8.5 Security Mechanisms in Cloud Computing

IoT applications can produce tremendous amount of data, hence the processing layer should have sufficient computing capability. Cloud computing is designed for this purpose. From this aspect, cloud computing plays the dominant role of IoT processing layer, hence the security techniques in cloud computing represent the mainstream of IoT processing layer security.

The security techniques in cloud computing include security protection on the cloud computing platforms, cloud computing services, and cloud data stored. Hence, a security architecture of cloud computing is proposed as depicted in Fig. 8.2.

8.5.1 *The Security Mechanism of Cloud Computing Platforms*

A cloud computing platform is composed of physical devices, networks, operating systems, and application software. To ensure the proper working condition of a cloud computing platform, the hardware should work with reasonable reliability. This can be measured by relevant industrial standards. The environment condition for the physical devices is also an important measure. Cloud computing platforms usually use distributed computing technology, where the physical devices are located in a number of distributed places. Every place with such devices should have proper security protection, including security guard, temperature control, and power supply. These security measures are about hardware protection.

While the hardware protection is important, security protection on software is also essential. A cloud computing platform is usually composed of a number of computing devices, each of such devices may have its own operating system, and there should be an overall operating system (called *Cloud Operating System* or *Cloud OS*) that integrates all those devices to work together, and sophisticated security mechanism is necessary.

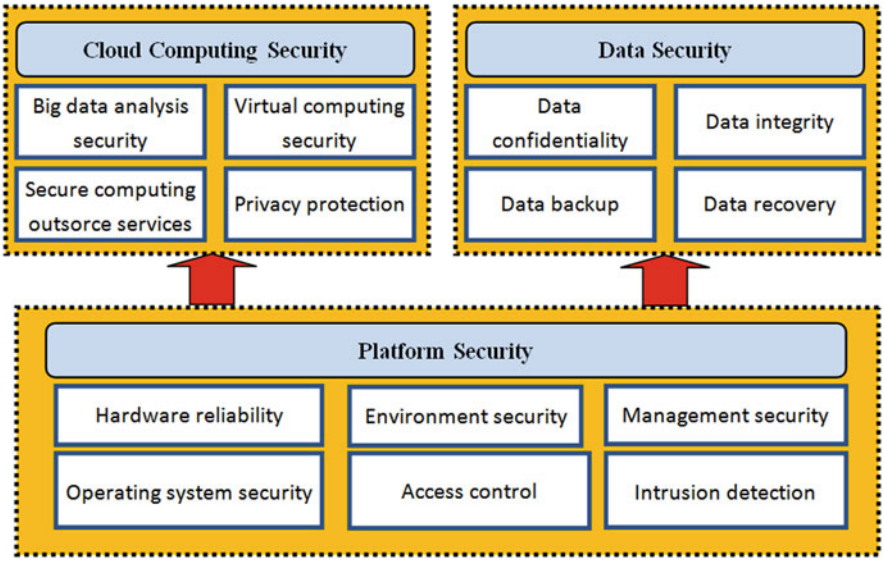

Fig. 8.2 Security architecture of cloud computing

Cloud computing platform is connected to the Internet, hence network attacks are a normal phenomenon. Hence, cloud computing platforms should be able to prevent system intrusion (e.g., using a intrusion prevention system, IPS), to detect system intrusion (e.g., using a intrusion detection system, IDS), to tolerant intrusion at certain level (i.e., the capability of intrusion tolerance), and to recover the system once it is damaged (i.e., the system recovery functionality).

As a common type of network attacks, DDoS attacks can hardly be prevented. However, the distributed architecture of cloud computing can effectively reduce the strength of DDoS attacks and mitigate the damage caused by DDoS attacks.

8.5.2 The Security Mechanism of Data Storage

Data storage is a simple task. However, when the amount of data is tremendously large, how to store the data is a problem. The problems with data storage in cloud computing environment include security as well as efficiency. With respect to the security measure, some confidential data should be stored in ciphertext form, and the associated problem is the management of encryption keys. With respect to the efficiency, it is to improve the efficiency when some data are used more often than others.

Data storage in cloud computing environment has to consider the data integrity problem. It is not simple problem of applying a message authentication code, but how to efficiently check the integrity of stored data, and how to enable users to check the integrity of their own data remotely. Note that the amount of data is large, some techniques have been designed that can avoid exhaustive computing.

8.5.3 The Security Mechanism of Virtual Computing

Apart from hardware and software, cloud computing makes good use of virtual computing technology, which provides much flexibility in dynamically managing the service limits for users. Virtual computing technology makes a physical computer into multiple virtual machines, each virtual machine can be configured separately with individual operating system, storage space, and application software. When the whole cloud computing resource is divided into different virtual computing environment, users can have their own preferred environment configuration. Virtual computing technology in cloud computing has a number of advantages, including the following:

Dynamic configuration. The resources of virtual machines can be dynamically changed if so needed. For example, A user may request some storage space for his/her virtual computing environment, and later on realize that the requested storage space is insufficient, then more storage space can be granted. This

flexibility particularly suits the industry users, since it is difficult to predict how much computing resources an enterprise needs in the future.

Efficient use of resources. The resources of cloud computing can be more efficiently used than individual computers, because the extra resources can be offered to more users.

High level of security. When multiple users share a same operating system, although users have their own rights, unauthorized access may be possible, even though the access control policy is perfectly applied, because users may set a wrong access permission on their own data by mistake. With virtual computing environment, unauthorized access becomes much harder, because cross boundary of virtual computing environment is difficult.

High reliability. With a physical computer, once it is severely damaged by a virus, e.g., a ransomware, it is likely to cause the crash of the whole system, unless the system is re-installed with a recovery backup, or is re-installed from scratch. However, with the virtual computing, the crash of a virtual machine can be rescued by the host machine, and the host machine has less network activities and hence has less chances of being affected by viruses.

Apart from the above, there are many other advantages that are not listed. Of cause, virtual computing also has some disadvantages, one obvious disadvantage is the overhead operation of the virtual machine. Fortunately, this limited overhead is negligible in powerful cloud computing environment.

There are some security problems with virtual computing. Typical security problems include the following:

1. *VM hopping*, which is the case when one virtual machine is able to monitor another virtual machine, or even the host machine. This is not a severe security threat, because even with physical machines, one machine may be able to remotely access another machine, and the mechanism with virtual computing is similar.
2. *VM escape*. If a user can penetrate the boundary of virtual computing environment, then the user may have the access right to the host machine like a Hypervisor, hence have the capability to make damage or shut down other virtual machines. However, this case is very rare in practice.
3. *Inherent network attacks.* By treating virtual computing environment as networked machines, many traditional network attacks also apply to virtual machines.

8.5.4 Security Mechanisms in Cloud Data

Another important functionality of cloud computing platforms is to provide data services. Such services include data storage, data encryption, data integrity protection, and data searching. Each of those services has some security problems.

Data storage. Data storage in cloud computing platforms is similar to traditional way of data storage, except the following characteristics:

1. Data users and data owners do not know where the data is physically stored, even the cloud computing operator may not know where the data is physically stored unless some tracking computation is done.
2. Data may be stored in multiple copies [11]. This is to prevent accidental damage of the data. If one copy is damaged, then a different copy is used. This makes a big difference from the traditional way of data storage.
3. Data may be split in parts, and different parts may be stored in different physical devices. This manner of storage is like secret sharing, particularly when the data content needs to be securely protected.

Data storage in cloud environment should also consider the efficiency of searching, because data may be stored in different physical devices. Moreover, searching encrypted data [4] is a special security requirement in cloud computing environment.

Data encryption. Data encryption can be done easily. However, with cloud data, encryption may be conducted only on parts of the data, because the cloud data is analyzed on the cloud computing platform. For example, data mining is a useful technology in database analysis. If the whole data is encrypted, then data analysis becomes hard. Note that the purpose of data encryption is only to protect the confidential part of the data, so partial data encryption can achieve the purpose of confidentiality protection, while keeping the data to be sufficiently useful to data analysis.

Data integrity protection. Message authentication code algorithms are effective methods for data integration protection. However, the process of verifying the data integrity status in cloud computing environment can be very different from traditional methods. In theory, data integrity verification is to re-produce the MAC tag and compare it with the recorded MAC tag. However, this process needs the complete data as input, which is costly for large amount of data. In cloud computing environment, integrity verification of data with a small fraction of the whole data is possible, this is called *provable storage*, it requires a knowledge protocol to judge whether the data stored in the cloud is complete [9].

Data searching. Data searching is a common database service. However, in cloud computing environment, data may be encrypted, the problem of data searching with encrypted data hence becomes a tough task. Some techniques of searchable encryption has been proposed to solve the problem [1, 2, 5, 8].

8.6 Summary

IoT processing layer is where large amount of IoT data are processed. A typical implementation of IoT processing layer is a cloud computing platform, where different security services are required.

This chapter briefly outlines the security threats and security services in IoT processing layer, including the differences of such security services with those for traditional information systems. In summary, the data confidentiality in IoT processing layer (e.g., a cloud computing platform) needs to consider the usability of the data, hence encryption of the whole data is not a wise way for data confidentiality service, because it would reduce the usability of the data in data analysis (e.g., data mining); The data integrity service in IoT processing layer needs to consider how the integrity can be verified without taking the whole data as input; The data storage service (which can be treated as part of data availability service) needs to consider how to make use of multiple copies of data, how to make data backup and conduct data recovery when needed. Although some of the security services in cloud computing environment are the same as that of traditional information systems, the implementation methods have big differences.

It should be noted that, cloud computing platform can provide other security computation services, including secure out-sourced computation [10], privacy-preserving computation [3, 6], and secure multi-party computation [13]. Those security computations have little to do with IoT applications, hence they are not discussed in this chapter, because they are not treated as important components in the IoT processing layer.

It should be pointed out that, data security and security computation services are challenging techniques in cloud computing, many of such security problems are considered as ongoing research topics.

References

1. K. Chamili, M.J. Nordin, M.J. Nordin, W. Ismail, A. Radman, Searchable encryption: a review. Int. J. Secur. Appl. **11**(12), 79–88 (2017)
2. B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan, Private information retrieval, in *Proceedings of 36th IEEE Conference on Foundations of Computer Science (FOCS)* (1995), pp. 41–50
3. J. Domingo-Ferrer, O. Farras, J. Ribes-Gonzalez, D. Sanchez, Privacy-preserving cloud computing on sensitive data: a survey of methods, products and challenges. Comput. Commun. **140–141**, 38–60 (2019)
4. Z. Fu, S. Xingming, Q. Liu, L. Zhou, J. Shu, Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans. Commun. **E98.B**(1), 190–200 (2015)
5. R. Handa, C.R. Krishna, N. Aggarwal, Searchable encryption: a survey on privacy-preserving search schemes on encrypted outsourced data. Concur. Comput. Pract. Exp. **31**(2), (2019)
6. F. Kerschbaum, Privacy-preserving computation, in *Privacy Technologies and Policy (APF 2012)*. Lecture Notes in Computer Science, vol. 8319 (Springer, Berlin, 2014), pp. 41–54
7. R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, Role-based access control models. IEEE Comput. **29**(2), 38–47 (1996)
8. D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in *Proceedings of 2000 IEEE Symposium on Security and Privacy (S&P 2000), Berkeley*, pp.44–55 (2000). https://doi.org/10.1109/SECPRI.2000.848445
9. J. Tian, X. Jing, Cloud data integrity verification scheme for associated tags. Comput. Secur. **95**, 101847 (2020). https://doi.org/10.1016/j.cose.2020.101847

10. Y. Yang, X. Huang, X. Liu, H. Cheng, J. Weng, X. Luo, V. Chang, A comprehensive survey on secure outsourced computation and its applications. IEEE Access **7**, 159426–159465 (2019). https://doi.org/10.1109/ACCESS.2019.2949782
11. M. Yi, J. Wei, L. Song, Efficient integrity verification of replicated data in cloud computing system. Comput. Secur. **65**, 202–212 (2017)
12. H.Y. Zhao, H.Y. Wang, H. Li, Analysis and improvement of the BLP security model. Adv. Mater. Res. **998–999**, 578–581 (2014)
13. C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, Y. Tan, Secure multi-party computation: theory, practice and applications, Inf. Sci. **476**, 357–372 (2019)

Chapter 9
Privacy Protection in IoT Applications

Privacy protection is a normal security requirement in network environment. The applications of IoT make the privacy protection issue more significant, because some IoT devices are used to catch personal private information such as health condition, medical figures, and location information. This chapter describes the concept and significance of privacy protection, introduces two kinds of privacy protection problems, i.e., identity privacy protection and location privacy protection. Some methods of identity privacy protection are described, including identity encryption, using a pseudonym, using temporary random identities, and other identity anonymity techniques. Some common methods for location privacy protection are described. It is pointed out that, privacy protection needs to be done properly, because the level of privacy protection and the usability of the privacy-preserved data have some conflicts, and they should be properly compromised.

9.1 Introduction

In our everyday life, annoying message interruptions happen from time to time, by means of phone call, SMS, social platforms, junk mails/emails, and paper notes (e.g., stickers). The main purpose of annoying messages is to broadcast advertisement, with certain level of fraud and cheating. In order to increase the effect of annoying messages, creators of such messages tend to classify target recipients to better meet their interests. How do the creators know what people's interests are? This is something to do with private information. For example, if someone knows the air ticket booking information, then cheating can be made more specific. In this sense, the ticket booking information is private information of travelers, and travel agencies are responsible for their customers' privacy protection.

The concept of privacy seems to be a common sense, however, the content of private information varies from person to person, and the content of privacy

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_9

protection varies in different applications. In general, privacy protection is to keep some personal information from getting into the hands of companies, hackers, government organizations, and other groups.

Privacy protection is a traditional security requirement in network environment. However, in the era of Internet of Things (IoT), private information becomes more sensitive, because IoT devices may affect the health and life of some patients (e.g., medical body sensors, artificial heart), which is far more important than data and information. The classification about what kind of information is private does not have a standard rule. When private information is connected to an identity, the identity information is the key for the data privacy, and this kind of problem is called *identity privacy*. For example, in a smart medical and healthcare system, the health information of a patient is something that needs to be provided to the doctors, and something to be kept confidential from the public. However, practically, large amount of medical records are stored in a data management center, which could be intruded by attackers (e.g., via virus affection), and such information can be stolen. In fact, medical records are just some facts, the connection to specific identities makes them to be private information. If the identity information is not connected to the medical record, then identity privacy problem does not exist. However, it is difficult to remove/hide the connection between a medical record and a specific identity, it is not as simple as simply removing/hiding the patient's identity information from the medical record, because other information may still contain sufficient information to converge to a specific identity for sure or with a high probability of being correct.

Considering the applications of smart medical and healthcare systems, identity privacy is an important security requirement for medical records, as well as the health status of people under the healthcare observation. Apart from the identity information, other information may also indicate the identity of the patients. A typical such information is the mobile phone number, which can usually identity a specific person, i.e., the owner of the mobile phone number. Other relevant personalized information includes living address, employer, age, gender, height, weight, and hobby. It seems that any single piece of such information may not be able to uniquely identity someone, however, the combination of such information may be able to do so.

More generally, any medical indicator contains some personalized information. Therefore, a complete privacy protection is to hide all such information from unauthorized access.

However, medical records are not just used for the patients themselves, but are also useful for medical analysis, which is important to diagnose and even to predict epidemics. So, hiding all the medical information is not a wise approach. This contradiction makes privacy protection difficult. It hence becomes a tradeoff problem as to what extent privacy protection should be provided.

By categorizing personal information into different categories, privacy can be categorized into different types. For example, Clarke presents four categories of privacy [4], which includes privacy of the person, privacy of personal data, privacy of personal behavior and privacy of personal communication. Finn et al. [6] further

extended the privacy into seven categories, they are named as privacy of the person, privacy of behavior and action, privacy of communication, privacy of data and image, privacy of thoughts and feelings, privacy of location and space, privacy of association. It is obvious that, all the types of privacy do not make sense if the identity information (in clear or being implied) is missing. So the privacy protection mechanism is to protect the identity information, i.e., to break the connection between the identity and other "private" information.

In IoT applications, a special type of private information is something connected to a specific location, this kind of private information is called *location privacy*. For example, in a smart transport system, location and the historical trace of some vehicles have location privacy. With respect to location privacy, to disconnect the location information from a specific identity is even harder, because the privacy problem is not simply about a specific identity, but to verify whether two or more identities are the same, although these identities may have already been protected (e.g., encrypted).

It should be noted that, the identity privacy is mostly concerned about personal information, while the location privacy can be about human or other things. If the location privacy is about human, then the location privacy is a special type of identity privacy. Common methods for identity privacy protection are to hide the information related to the identity, and alternative methods for location privacy protection can be done by hiding the location information or by generating a large number of fake location data to confuse the real location data, depending on the specific applications.

Privacy protection is not a simple technical problem, it has something to do with social security, law and politics, and even the national security. This chapter only considers some technical aspects of privacy protection, and practical solution toward privacy protection also involves social, political, and legal factors.

9.2 Privacy Protection by Identity Anonymity

There are many different methods for identity privacy protection. An effective method is identity anonymity, which is to hide the real identity. A model to anonymize the identity information can be depicted in Fig. 9.1.

The identity anonymity seems to be easily achieved by simply removing the identity information. However, this does not fit the data format of many practical applications. Filling the field of identity card number by a random number may or may not be working, because removing the identity card number may cause the data to be useless, or other parts of the data still contain some information about the identity.

An intuitive approach of identity anonymity is to use pseudonyms. A pseudonym does not have obvious relation to the real identity, and in many practical applications, there is a mechanism to link the pseudonym to the real identity, which is managed by an authentication center. The relations between pseudonyms and

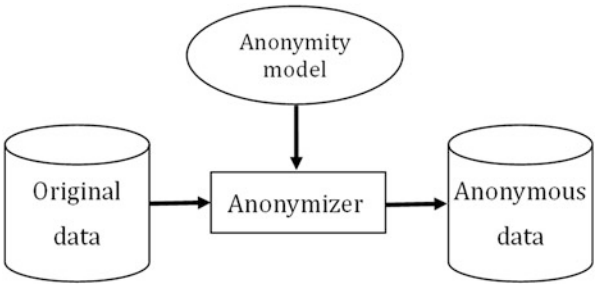

Fig. 9.1 An identity anonymization model

their real identities are managed by an authority, which can be the verifier, or a trusted third party. The Temporary Mobile Subscriber Identity (TMSI) used in mobile communication systems is a pseudonym, which is linked to the real identity (the IMSI) of the subscriber. Although the TMSI changes from time to time, their corresponding real identity, i.e., the IMSI, remains the same.

In 1985, Chaum proposed an anonymous credential system [1, 2],where an Issuer generates anonymous credentials to a users, who will be the prover to prove the authenticity of the anonymous credential provided to someone, who acts as a verifier to conduct the authenticity verification. The anonymous credentials also look like pseudonyms, and their authenticity can be verified.

9.2.1 Group Signatures for Limited Identity Anonymity

Apart from the pseudonym techniques, there are other techniques to anonymize the identities. One of such a technique is group signature. The concept of group signature was propose by Chaum and Heyst in 1991 [3]. A group signature is a kind of signature algorithm that produces a group signature on a specific message, and the validity of the group signature can be publically verified. A group signature system needs to have a group manager who issues signing keys to individual group members, and has the following properties:

1. Any group member can produce a valid group signature using his individual key, and no one outside the group is able to compute a valid group signature on any message;
2. The validity of the group signature can be verified using the group public key, and the process of signature verification does not reveal which group member generated the group signature;
3. In case of dispute, the signature can be "opened" to reveal the identity of the signer. The process of opening the group signature may or may not need the collaboration of other group members.

A simple instance of group signature can be briefly described as follows:

System setup: Group manager Z generates kn pairs of public keys and private keys (pk_i, sk_i), $i = 1, 2, \ldots, kn$ for n group members, each group member hence has k private keys. The group manager Z publishes all the public keys as the group key.

Signing: When group member g_i wants to create a group signature on message m, he uses one of his private keys to compute the signature $s = Sign(sk_j, m)$, and publishes (pk_j, s) as the group signature.

Verification: When one wants to validate the group signature, he first checks whether pk_j belongs to the claimed group. This can be done by checking the public key of the group announced by the group manager Z earlier. Then he verifies the validity of the signature s using the public key pk_j.

The anonymity of the signer is ensured since no one knows which group member has the private key corresponding to the public key pk_j, except the group manager Z. Note that if the group member uses the same public key to create a signature on a different message m_2, although the verifier does not know who signed the message m_2, he knows that it must be the one who signed the message m. In order to maintain the complete anonymity, any private key cannot be used to sign more than one message, hence, any group member is restricted to create up to k different group signatures.

The existence of the group manager is not a necessity. In the case where the group manager is not deemed, it is equivalent to the case when the role of the group manager is taken by the cooperation of the group members.

The group signatures are a generalization of the credential mechanisms, in which one person shows his authenticity by proving that he belongs to a certain group, while the process of authenticity does not reveal his real identity. This provides certain level of identity privacy, because the group signature reveals the range of the identity, i.e., the signer must be a member of the claimed group.

9.2.2 Ring Signatures for Limited Identity Anonymity

Another technique with group-based anonymity is ring signature. The concept of ring signature was proposed by Rivest et al. [9, 10]. The original concept was proposed to function as a way to leak secret information, without actually revealing who signed the message. A scenario of such is from high ranking government officials, who wants to confirm that the message is really from the government officials, and does not want to reveal which official member signed the message.

A ring signature is also a type of digital signature that can be performed by any member of a group of users, and can be publicly verified as being signed by a group member, and the signature does not reveal which group member generated the signature. Different from group signature, a ring signature system does not need a group manager, hence the actual signer cannot be revealed. In fact, a ring signature

is created by the signer without any cooperation of other group members, and the group can be freely defined at the signer's discretion, provided that the public keys of the group members are known.

A typical example of ring signature given in [9] based on RSA public key system can be described below. Let a group of r members be defined, and the i-th member has a public key $P_i = (e_i, n_i)$, which correspond to RSA encryption

$$y_i = x^{e_i} \pmod{n_i}.$$

For the convenience of writing, a function is defined as

$$C_{k,v}(y_1, y_2, \ldots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus \ldots \oplus E_k(y_1 \oplus v)))$$

where E_k is a symmetric encryption algorithm with $k = H(m)$ as the encryption key, m is the message to be signed, and $H(.)$ is a collision resistant hash function. When someone wants to create a ring signature on message m, he defines a group with public keys (e_i, n_i), $i = 1, 2, \ldots, r$, where his public key is (e_s, n_s), $1 \leq s \leq r$. Then the signing process can be done by the following steps:

1. Choose a random number v of appropriate size. Choose random numbers x_i, $1 \leq i \leq r, i \neq s$, and compute $y_i = x_i^{e_i} \bmod n_i$.
2. Solve the following equation for y_s:

$$E_k(y_r \oplus E_k(y_{r-1} \oplus \ldots \oplus E_k(y_1 \oplus v))) = v.$$

   When the public encryptions are properly handled, they are permutations, and the above equation must have a unique solution y_s, which can be solved using and RSA decryption algorithm.
3. Compute x_s such that $y_s = x_s^{e_s} \bmod n_s$ holds. This is the RSA decryption process, and can be done easily given the private key d_s.
4. Output the ring signature which is a $(2r + 1)$-tuple:

$$(P_1, P_2, \ldots, P_r, v, x_1, x_2, \ldots, x_r, y_1, y_2, \ldots, y_r)$$

   together with message m, where P_i is the public key information of the i-th group member.

The signature verification process is simple. On receiving the message and the ring signature, the verifier computes $k = H(m)$, verifies whether y_i is indeed the signature of x_i using the public key of the i-th member. As for the above protocol, it is to verify whether equalities $y_i = x_i^{e_i} \bmod n_i, i = 1, 2, \ldots, r$ all hold. In fact, it is not necessary to include $y_1, y_2, \ldots, y_r$ in the ring signature, because the verifier can compute $y_i = x_i^{e_i} \bmod n_i, i = 1, 2, \ldots, r$. Then the verifier checks whether equality

$$E_k(y_r \oplus E_k(y_{r-1} \oplus \ldots \oplus E_k(y_1 \oplus v))) = v$$

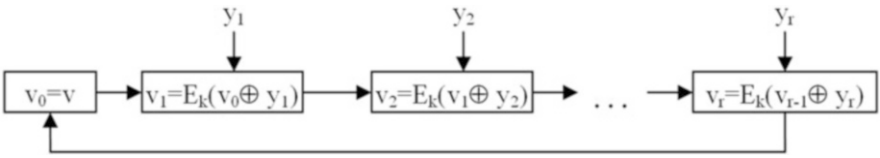

Fig. 9.2 The behavior of a ring signature

holds. If so, the ring signature is accepted. It is noted that some tips have been mentioned in [9] to ensure that the public encryptions are permutations, this is to avoid the cases when the modulus operation evaporates some information.

From the verification process of a ring signature it can be seen that, the process starts from a random number v, which is XOR'd with y_1 and the result is encrypted using a symmetric encryption algorithm E with $k = H(m)$ as the encryption key, the output is then XOR'd with y_2, and the output is then encrypted with E. This process goes sequentially till y_r, and the final output of the symmetric encryption algorithm E is magically back to v, the original random number. This behavior looks like a ring, as shown in Fig. 9.2, so it is called a ring signature.

In terms of privacy, ring signatures provide higher level privacy than group signatures in the sense that the identity of the signer cannot be revealed, and the price is that, the singer may maliciously or irresponsibly generate ring signatures on debatable or even malicious messages. Practically, such concerns can be amended by other management mechanisms, e.g., by grouping people with same level of credits together.

9.2.3 Some Practical Tools for Privacy Protection

There are a number of anonymous authentication tools available for different purposes. The most commonly used ones include Idemix, PrimeLife, and U-Prove.

The Idemix

Identity Mixer (Idemix) was proposed by IBM for identity privacy protection, while enabling verification of the authenticity. It allows users to authenticate anonymously. Instead of providing their identity, users can obtain access to a resource by merely proving their possession of the required credentials without revealing the credentials themselves. In Idemix, any user is assigned an electronic identity (eID), enabling anonymous authentication by providing the credentials of the eID, instead of the real identity. The process of anonymous authentication of Idemix can be depicted in Fig. 9.3.

The cryptographic library of Idemix offers the most common cryptographic algorithms to achieved such anonymous authentication. This comprises the functionality for the issuer, client, and service provider.

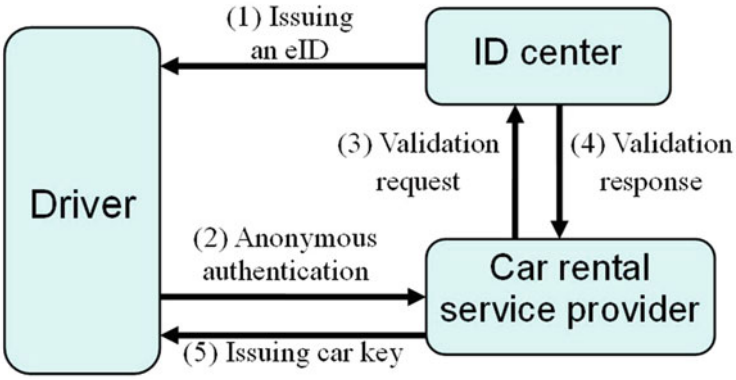

Fig. 9.3 The anonymous authentication process using Idemix

PrimeLife

In order to provide individuals in the information society the capability to protect their autonomy and retain control over personal information, IBM's Zurich Research Laboratory leaded 14 other partners finished a project funded by Europe and resulted in PrimeLife.

PrimeLife claims to bring sustainable privacy and identity management to future networks and services, protect privacy in emerging Internet applications such as collaborative scenarios and virtual communities. In addition, PrimeLife solves the problem about how to maintain life-long privacy.

U-Prove

In the network applications, payment can be made over the Internet. Such a payment needs to provide sufficient personal information as well as the information of a credit card, which will be verified by the credit card issuer or a financial body. However, the process reveals so much information enabling the payee to masquerade the payer for other purposes. Although practically, the payee is usually a large company or service provider with trustable business goodwill, the personal information could be stolen by network hackers, and the hackers can use the personal information to masquerade the payer for nefarious purposes. This is not a hypothetic concern, relevant financial frauds happen frequently and large amount of personal information has been illegally compromised.

The U-Prove system is designed to enable secure transactions without giving third parties the ability to masquerade as the user—the payer, and enable the user visit websites and make purchases without those sites being able to track the user. It is also difficult to combine different pieces of information to draw a more complete picture of the user.

9.3 Privacy Protection Based on Data Linkage

Private information includes names, identity numbers, mobile phone numbers, residential addresses, etc. However, each piece of information is not private, only when different pieces of information are linked together, they become private. So the privacy protection is actually to protect the linkage among different pieces of information.

Having the sense of privacy protection, people tend to protect their personal information as much as possible. However, in many cases, personal information does not seem so private. For example, the name of someone in a party may not be protected, because revealing such information does not cause any problem. Assuming that the person went to a golf course with a couple of friends after the party, where their names are published on a web site. One of them had a traffic accident on the way going home from the golf course. When the accident is reported, the name of the person is anonymized for privacy purposes, but the description about where he came from (from the golf course to his home) and why he had some alcohol in his blood (the party before the golf course) was reported. Each message does not seem to have privacy problem, however, when the different descriptions are put together, the identity of the person can be clearly recognized. This privacy leakage is caused by information integration, or data linkage.

Data mining is a mature technology to reveal group behaviors by analyzing large amount of data linkage. However, privacy mining is different from data mining in the sense of individuals instead of for groups. When data from different databases are analyzed, data can be visualized by graphs, with different fields of data as points, and their linkages as edges, then it is possible to merge two graphs together if a common and unique identifier (e.g., a mobile phone number, or an identity number) can be found, which forms a larger graph. When the graph is large enough that links the identity information and some sensitive information (e.g., medical record), then the privacy is revealed. An example of graphical behavior of privacy mining using the data linkage can be depicted in Fig. 9.4.

As shown in Fig. 9.4a and b both hide the identity information X' and Y', they have no sub-graphs in common, except the sub-graphs $(E - I, E - H)$ and $(E' - I', E' - H')$, $(D - K)$ and $(D' - K)$, $(C - L)$ and $(C' - L')$, that have some similarities, where the alphabet with and without a prime means that they represent the same kind of parameters but not unique. The similarity gives a possibility that $E = E'$, and the possibility that $D = D'$ is also high due to they have a same node K. Hence, there is a good possibility that $X = Y$ holds. By noticing that figure (c) can be judged as a sub-graph of figure (a), although with some possibility of exception, it can be concluded that the relations of the nodes can be shown in figure (d). This conclusion is not 100% correct, and with the help of more graphs, the conclusion that $X = Y = Alice$ can be made more confident. This is just a simple principle description, more deterministic deduction involves numerical assignments and deduction rules.

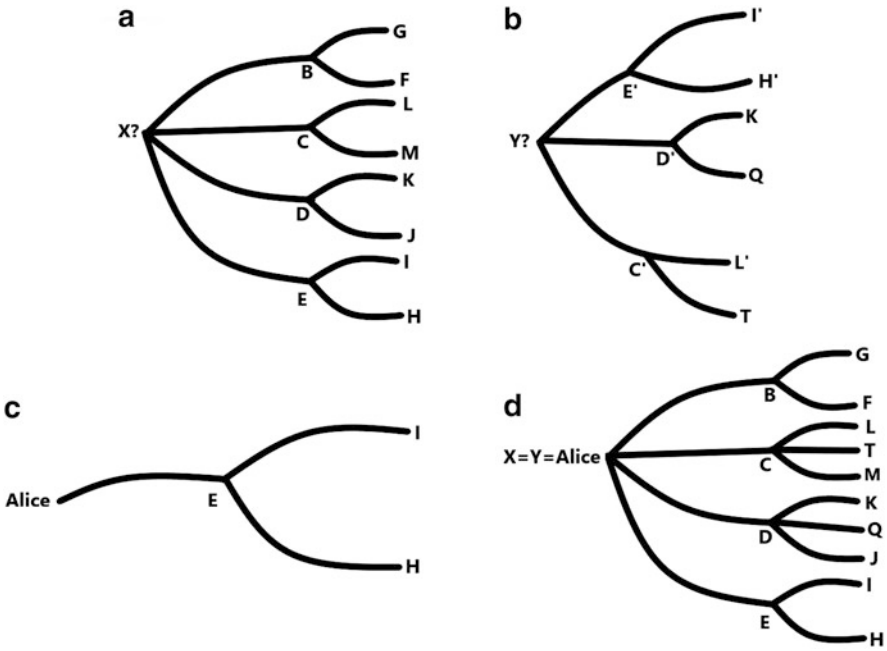

Fig. 9.4 An example of graphical behavior of privacy mining based on data linkage

However, how to thwart this privacy mining is a challenging problem, because there is no standard model to formulate what kind of personal information can be made publicly available. It is impractical not to record the personal information in all the applications. For example, in an on-line shopping application, the address information must be made clear, so that the purchased goods can be delivered to the correct address. For security reasons, a real identity must also be provided.

A general mechanism to reduce the chances of privacy mining based on different databases is that, the personal information that can uniquely (or almost uniquely) distinguish the identity should be hidden once the recorded data becomes a historical data. For example, once an on-line purchasing is completed, the purchasing information should be anonymized. Hence, data anonymization [5, 8] is a relevant technique against privacy leakage/disclosure. When the data to be anonymized is an identity, the technique of anonymization is called *de-identification* [7]. De-identification removes identifying information from a dataset so that individual data cannot be linked with specific individuals. Another typical data anonymization technique is called k-anonymity [12], which means that each combination of data values of the identifier attributes is shared by at least k records, so that getting any less than k records will not be able to recover the relevant data values.

Nevertheless, large amount of data are used for further analysis, and data mining is a common data analysis method. When the data with private information is used for further analysis, including data mining, private information needs to be removed

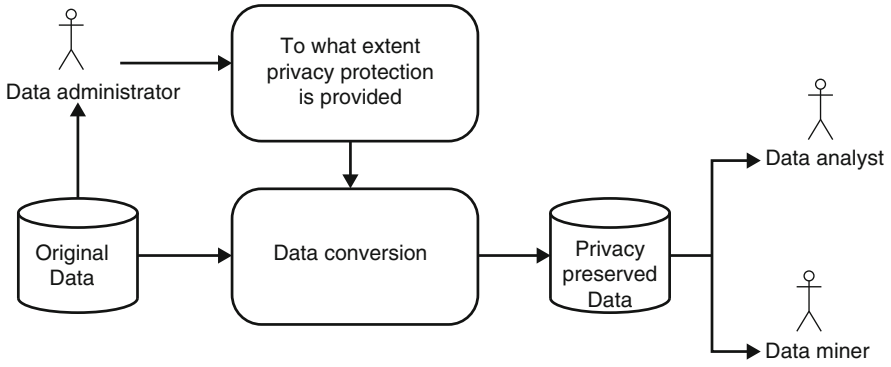

Fig. 9.5 Data preparation for privacy-preserving data mining

as much as possible. When the private information is supposed to be removed, the data mining based on such data is called *privacy-preserving data mining*. The concept of privacy-preserving data mining was proposed in [11], which applies the techniques of data mining and privacy protection together. The process of data preparation for privacy-preserving data mining can be depicted in Fig. 9.5.

9.4 Location Privacy Protection

Parents may want to know whether their kids are at home, in the kindergarten, or playing in a garden. Such information is related to location. However, parents do not want others know the location of their kids. The way to hide the location information from unauthorized users is a location privacy problem.

Another scenario of location privacy is traffic information, users report where they are and how fast they drive, so that the traffic condition can be reflected back to users using a live traffic navigation system. Although the location information is provided precisely, the identities of the users who report their location should be anonymized. This is similar to the problem of identity anonymity. Since the data is about location information, it is called location privacy protection.

9.4.1 Location Privacy Protection by Hiding the Location Information

Location information is important in many applications, such as location-based service (LBS). The LBS is usually provided based on a geographic information system (GIS), and the GIS use the positioning technology. Typical positioning technologies include Global Positioning System (GPS), BeiDou Navigation Satellite

System (BNSS), Wi-Fi and Cell-id based positioning systems (NGPS), and mobile positioning systems.

The LBS is closely related to the location of users. For example, the navigation system needs to know the location of a user so that a regional map showing the navigation route can be displayed properly. The LBS can provide convenience to people's life, particularly provide assistance to travel related services, however, the location privacy of users can be a security concern. It is obvious that users should provide their location information to the LBS provider so that services can be provided, this process enables the LBS provider to have the location information of the users. If an attacker accesses the system of the LBS provider, then the attacker is able to know the users' location as well as their moving path, which violates the users' location privacy.

A typical location privacy violation can be described as follows: User U sends message M to LBS provider, which includes location information L of U. If an attacker A gets the location information L, then the attacker can disclose the location of the user in different ways, including the following:

Discloser of location. If attacker A knows that the location information L belongs to user U, then A can conclude that L is the actual location of user U. For example, if the location information L is inside a private grape land, then the user is likely to be the owner of the grape land, or a family member of the land owner. This is a consequence caused by the leakage of such location privacy.

Discloser of moving path. If attacker A gets to know a collection of location information $L_i, i = 1, 2, \ldots, n$ of user U, then A is able to know a path that U has gone through, which can be used to predict where U is likely to be next.

Discloser of location-related information. If attacker A gets the location information L which is a special place, then the status of user U may be disclosed with large possibility. For example, if the location L is a hospital, then user U may be ill, or may be with a patient. If the same location information repeats with a relatively long time range, then user U is likely ill and is being treated in the hospital, unless U works in the hospital.

Although the location information itself makes little sense, when the location information is connected to an identity, then it becomes private information. Location privacy protection is meant to protect the unauthorized disclosure of connection between an identity and the location information at a specific time. While protecting the discloser of the location information is a way of location privacy protection. Alternatively, the location privacy can be protected by hiding the identity information.

9.4.2 *Location Privacy Protection by Hiding the Identity Information*

If the identity of a location can be hidden, then the connection between the location information and the identity information is broken, hence the location privacy can be protected.

Intuitively, an identity can easily be hidden. For example, when providing the location information, no identity information is provided. However, authentication is usually required to eliminate the fraud messages, without identity information, authentication cannot be done, even anonymous authentication requires some data related to identity, which is a proof about the authenticity of the identity. This shows that the identity information cannot be simply removed. Another approach is to use a fake identity to replace the real identity. However, the identity information needs to be used for authentication, and a fake identity cannot pass the authentication process. A practical approach of hiding the identity information is to use a random number as a temporary identity, as used by the mobile communication systems. When the real identity is used to pass the authentication, a temporary identity is issued. This temporary identity is used for later authentication, and the temporary identity can be updated when successful authentication is conducted. In mobile communication systems, the process of TMSI reallocation is composed of two message exchanges: the network sends the user equipment a TMSI reallocation command, and the user equipment sends back a TMSI Reallocation Complete message as the response to the network, see Fig. 9.6.

However, when the mobile user roams to a different location, the original identity of the mobile equipment must be used. This mechanism of using the random temporary identities can provide certain level of identity privacy, making it difficult for the attackers to trace a mobile user.

The purpose of changing the temporary identity is to avoid possible tracing. In the privacy protection of RFID tags, the same mechanism is used, where the pseudonyms of an RFID tag keeps changing every time it is requested for identity authentication. Otherwise, the tag can be tracked, even if its identity is encrypted, if the encrypted identity is a static ciphertext.

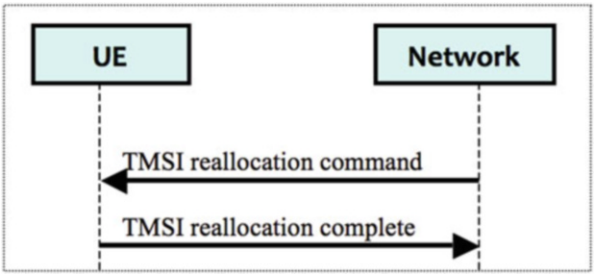

Fig. 9.6 The TMSI reallocation process in mobile communication networks

9.4.3 Location Privacy Protection by Fake Location Information

Study on endangered and compelling animals tending to better protect them often sticks GPS devices to them, so that the location and path of moving can be tracked. However, malicious hunters are keen to know the location of such animals. If the location information gets into the hands of malicious hunters, then it would increase the death threat of the animals. While modifying the GPS information (e.g., by an encryption algorithm) can protect the location information, however, most of the GPS devices do not provide the functionality of data encryption. In order to prevent malicious hunters from getting the location information of the animals being studied, an alternative approach is to transmit a large amount of fake location data, when the fake location data is overwhelmingly large, the malicious hunters cannot differentiate whether the location information is real or fake, hence would lost interest to getting such information.

9.4.4 Location Privacy Protection by Fuzzy Location Information

Another method to hide the location information is to use a fuzzy location instead of the precise location. The fuzzy location is an area that contains certain number of objects, so that the object cannot be uniquely identified. As shown in Fig. 9.7, object N_1 at location (x_1, y_1) and object N_3 at location (x_3, y_3) are both in a field containing 6 objects. Instead of sending their precise location, object N_1 and object N_3 can send an area to the server, the area can be a square, a circle, or other regular forms.

In order to avoid the cases where an area has only one object, so that the identity and the precise location of the object can be identified, the idea of k-anonymity technique can be used here. The k-anonymity means that, when extending the

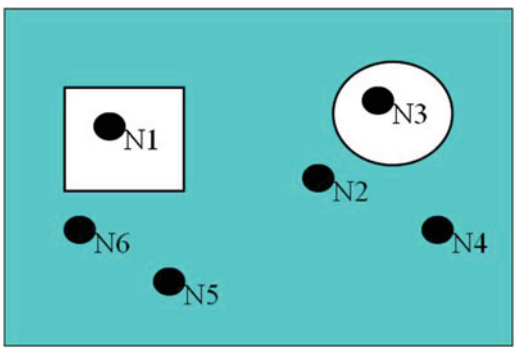

Fig. 9.7 Location privacy protection by fuzzy location information

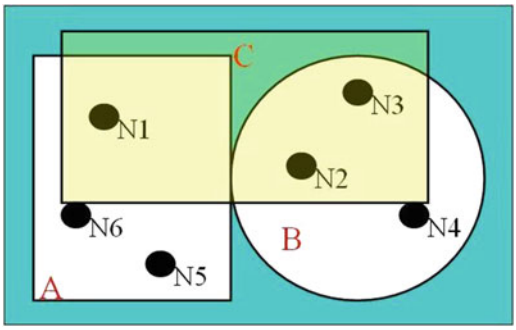

Fig. 9.8 Location privacy protection by combining the fuzzy location information and k-anonymity

location of some object into an area, it should ensure that the area has at least k users all together, including the object itself. For example, by employing the technique of 3-anonymity, when the location of object N_1 is extended into a rectangle area, the area must contain at least 3 objects. In Fig. 9.8, area A can be used as the fuzzy location of N_1, because it contains three objects including N_1. Similarly, area B can be as the fuzzy location of N_3, because it contains three objects including N_3. It is also noted that area C can be used as the extended area of the fuzzy location of both object N_1 and object N_3. However, it is obvious that this shared location leaks more location information than using separate areas, if the fuzzy location of both N_1 and that of N_3 are reported.

It can be seen that, the fuzzy technique provides limited location privacy, which is useful in applications when the rough location of some objects is useful (e.g., a kid in a specific kindergarten), and the precise location is not needed and should be protected.

9.5 Graded Privacy Protection

Privacy protection becomes more and more important in the IoT and big data era. However, there is a barrier between the privacy protection and the usability of data after the process of privacy preserving. It is noted that the higher level of security protection, the lower level of usability of the privacy-preserved data. Given that, it is difficult to make a privacy protection method to be universally applicable.

When data is stored into a database, usually multiple parameters are used to store different parts of the data, each parameter may have a different feature. Some of the parameters are closely associated with private information, and should be protected for the purpose of privacy protection, while other parts of the data are not so sensitive. Note that the nature of privacy is not a certainty issue, it is a possibility. For example, the identity of a user may or may not be private information, depending on what the identity is linked to. The height, weight, and the body mass index may not be private information within an organization, and would become private to the public.

Table 9.1 Privacy protection level versus the usability of data

$t := 0$	Identity	Contact number	Suburb	Blood pressure	Body temperature	Doctor's advice
0.2	X	Contact number	Suburb	Blood pressure	Body temperature	Doctor's advice
0.5	X	X	Suburb	Blood pressure	Body temperature	Doctor's advice
0.6	X	X	X	Blood pressure	Body temperature	Doctor's advice
0.8	X	X	X	X	Body temperature	Doctor's advice
0.9	X	X	X	X	X	Doctor's advice

Given the above notification, it is hence proposed that graded privacy protection would be more flexible in adjusting the tradeoff between the level of privacy protection and the usability of the privacy-preserved data. In graded privacy protection, different parts corresponding to different database parameters are labeled with different grade values, reflecting the weight of privacy, the larger the value is, the more important that part of data is treated in terms of privacy protection. For example, considering a database of medical records, where the identity parameter is marked as 1, the mobile phone number is marked as 0.8, the suburb is marked as 0.6, the blood pressure is marked as 0.5, the body temperature is marked as 0.2, the advice made by doctors is marked as 0, then privacy protection can target at level t, i.e., all the parameters with privacy grade value larger than or equal to t should be protected. Then different values of privacy level will result in different usability of the data, as shown in Table 9.1.

From Table 9.1 it can be seen that, when the privacy protection level is 0.5, the privacy protection is made fairly well, because the identity and contact number fields are protected, while the data is still useful for the purpose of medical data analysis. When the privacy protection level gets higher, the data becomes less useful.

The purpose of graded privacy protection is to apply different security protection methods on different data parameters, so that different level of privacy protection can be released for analysis. For example, if the data parameters with certain privacy grades are encrypted with different keys, then different capability of decryption gets data with different level of privacy protection.

The graded privacy protection is a topic that needs further study.

9.6 Summary

Privacy protection is a special security requirement in many network-based applications. The most common privacy issues are identity privacy and location privacy. The identity privacy is about protecting the identity information that is linked to

some personal data, such as medical record, and the location privacy is about breaking the linkage between an identity and a location, practically the linkage of identity information and the location information should be associated with a specific time.

With respect to the identity privacy protection, it is to hide the information (personal information) that can be used to identity a specific person, not necessarily the information of the identity itself. With respect to the location privacy protection, two approaches are introduced, they are location privacy protection by protecting the identity, and that by protecting the location information.

When private information is properly protected, the data is called privacy-preserved data, such data is useful for public analysis. Data mining based on such data is called privacy-preserving data mining. However, different pieces of data may have a same node, the common node in different data sets can be used to link the data sets together to form a larger data graph, and private information has more chances to be revealed in the larger data graph. This finding based on data linkage is a privacy mining problem. Privacy protection and privacy mining are conflicting techniques, it is suggested that, data privacy protection should be done at an appropriate level, so that the data is useful for data analysis (e.g., privacy-preserving data mining), while privacy mining can hardly find the private information.

Considering that privacy protection should be done at a proper level of security, too much privacy protection would make the data less useful, while too week privacy protection is vulnerable to privacy mining, the concept of graded privacy mining is hence introduced, which can be used to better balance the tradeoff between the level of privacy protection and the usability of data. This topic needs further study.

References

1. D. Chaum, Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (1985)
2. D. Chaum, Showing credentials without identification, in *Advances in Cryptology – EUROCRYPT'85*, ed. by F. Pichler. Lecture Notes in Computer Science, vol. 219 (Springer, Berlin, 1985), pp. 241–244
3. D. Chaum, E. van Heyst, Group signatures, in *Advances in Cryptology – EUROCRYPT'91*, ed. by D.W. Davies. Lecture Notes in Computer Science, vol. 547 (Springer, Berlin, 1991), pp. 257–265
4. R. Clarke, Introduction to Dataveillance and information privacy, and definitions of terms, Xamax Consultancy, Aug 1997. http://www.rogerclarke.com/DV/Intro.html
5. J. Domingo-Ferrer, J. Soria-Comas, Data anonymization, in *Proceedings of International Conference on Risks & Security of Internet & Systems (CRiSIS 2014)*. Lecture Notes in Computer Science, vol. 8924 (Springer, Berlin, 2015), pp. 267–271
6. R.L. Finn, D. Wright, M. Friedewald, Seven types of privacy, in *European Data Protection: Coming of Age*, ed. by S. Gutwirth et al. (Springer, Berlin, 2013), pp. 3–32
7. S.L. Garfinkel, De-identification of personal information. National Institute of Standards and Technology Internal Report 8053, Oct 2015. https://doi.org/10.6028/NIST.IR.8053
8. F. Kohlmayer, F. Prasser, C. Eckert et al., A flexible approach to distributed data anonymization. J. Biomed. Inform. **50**(8), 62–76 (2013)

9. R.R. Rivest, A. Shamir, Y. Tauman, How to leak a secret, in *Advances in Cryptology - ASIACRYPT 2001*, ed. by C. Boyd. Lecture Notes in Computer Science, vol. 2248 (Springer, Berlin, 2001), pp. 552–565
10. R.R. Rivest, A. Shamir, Y. Tauman, How to leak a secret: theory and applications of ring signatures, in *Theoretical Computer Science, Essays in Memory of Shimon Even*. Lecture Notes in Computer Science, vol. 3895 (Springer, Berlin, 2006), pp. 164–186
11. R. Shilpa, D. Patel, Survey on privacy preserving data mining techniques. Int. J. Eng. Res. Technol. **9**(6), (2020). Published (First Online), 26 June 2020. https://doi.org/10.17577/IJERTV9IS060568
12. L. Sweeney, *k*-anonymity: a model for protecting privacy. Int. J. Uncertainty Fuzziness Knowledge Based Syst. **10**(5), 557–570 (2002)

Chapter 10
RFID System Security

This chapter introduces some typical security threats in RFID systems, including RFID tag cloning, RFID tag tracking, and relay attacks. Tag cloning may cause spoofing attacks, tag tracing violates the location privacy of the thing/person that the traced tag is attached to, and relay attack can link a genuine RFID tag to an authorized reader, even though the tag may be physically far away from the reader. Some techniques to thwart those attacks are described, including cryptographic methods, hardware techniques, and specific security protocols.

10.1 Introduction

In order to differentiate and record physical things (i.e., material assets), things are labeled with different identities. The identities are generated by a coding method. Typical such coding methods include bar codes, two-dimensional codes (e.g., QR codes), and Radio Frequency Identification (RFID) tags. The bar codes and QR codes are visible and represent static identities, while the codes represented by RFID tags are invisible to human eyes, they can be read using specific devices (RFID readers). The process to read an RFID tag needs to follow a special communication protocol that is executed between an RFID tag and a reader.

RFID is a technology that uses electronic tags to store identities electronically, which are digital strings of specific format. The electronic RFID tags are small chips that can be attached to an object or even implanted inside the object, hence the object can be identified by the identity of the RFID tag. The tags contain information that can be transmitted to a reader via radio waves, so that the reader and the tag do not need to have physical contact.

There are two main types of RFID tags. The most popular type of RFID tags do not have power supply, they need to be exposed to an electronic magnetic field to get energy by electromagnetic conversion. Such RFID tags are called *passive RFID*

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_10

tags. Another type of RFID tags are battery-powered, they are called *active RFID tags*.

Passive RFID tags use three main frequencies to transmit data: 125–134 kHz, known as Low Frequency (LF); 13.56 MHz, known as High Frequency (HF) and Near-Field Communication (NFC); and 865–960 MHz, known as Ultra High Frequency (UHF). Different frequencies affect the ranges that the RFID tags can communicate with a reader. However, the range of passive RFID tags should not be very large, because the RFID tags need to be exposed to the electromagnetic field of the reader to get energy.

There are two main types of passive RFID tags: inlays and hard tags. Inlays are typically quite thin and can be stuck on various materials, whereas hard tags are made of a hard, durable material such as plastic or metal.

Active RFID tags use two main frequencies to transmit data, i.e., 433 MHz or 915 MHz. Since RFID tags are usually small in size, the batteries in active RFID tags are usually not replaceable, so the batteries are expected to supply enough power for the RFID tags to last for reasonably long time, practically a couple of years.

There are two main kinds of active RFID tags: beacons and transponders. Beacons send out a ping message every few seconds, and their signal can be readable from hundreds of meters away in an open field, which consumes much energy. Transponders require the use of a reader to transmit data. When a transponder RFID tag is within the range of a reader, the reader first sends out a signal to the transponder, which then pings back with the relevant response. Because transponders only activate when they are near a reader, transponders are much more battery-efficient than beacons.

When being exposed to a radio wave of the correct frequency, an RFID tag broadcasts its identity information (e.g., the EPC number) to the RFID reader, enabling the RFID reader to get the identity information of the RFID tag. Some RFID tags have integrated with other kinds of data such as an account balance (e.g., an electronic bus ticket), where the balance is kept in the RFID tag and is deducted every time it communicates with a reader successfully, while the RFID reader collects the information and will update the database in a data processing center. The main difference between RFID tags and other kinds of codes for identification is that, the RFID tags have computation and communication capabilities, so that some mechanism to protect the identity of RFID tags is possible. This special feature of RFID tags makes the RFID security techniques significant.

The RFID systems have widely been used in different areas, including retailers, logistics, transport, entrance access (e.g., hotel room keys), and name-tags.

10.2 Introduction of RFID Systems

As mentioned above, there are different types of RFID tags, and they suit different applications. In general, RFID tags are devices representing unique identities, those identities can be globally unique, or unique in a specific application system.

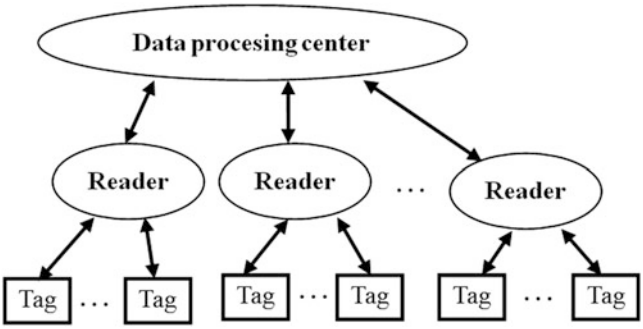

Fig. 10.1 An architecture of an RFID application system

An RFID system is composed of RFID tags, some readers, and a data center that processes the data. The identity itself is just a string, it makes no sense unless when the identity is associated with other useful information. When an RFID tag is stuck to a certain kind of goods, a data processing center records the description about the goods, and the description information is associated with the identity information of the RFID tag. When a reader gets the identity information from an RFID tag, it transmits the identity information to the data processing center, which then finds the relevant information associated to the identity. Readers also have their identities, and the data processing center can identify which reader sends the identity information, and can do corresponding processing, e.g., to show the status of the goods, or the price of the goods, and update records whenever necessary. The architecture of an RFID system can be depicted in Fig. 10.1.

From Fig. 10.1 it can be seen that, a reader can communicate with multiple RFID tags, and the data processing center can communicate with multiple readers. It should be noted that, the communication between an RFID tag and a reader is wireless, while the communication between the data processing center and a reader can be wireless (e.g., via a mobile communication network) or wired. RFID readers can be mobile or fixed. For convenience, the mobile readers usually use wireless communication with the data processing center, and fixed readers usually use wired communication. Although conventionally they are called readers, some of the applications require the readers to be able to update some data stored in RFID tags, in such case, the RFID readers are also RFID writers.

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) has worked out some standards, known as ISO/IEC 18000 series. The series is composed of six relevant parts, their names are as follows:

1. "Information technology—Radio frequency identification for item management—Part 1: Reference architecture and definition of parameters to be standardized", it defines the parameters to be determined in any standardized air interface definition in the subsequent parts of ISO/IEC 18000;

2. "Information technology—Radio frequency identification for item management—Part 2: Parameters for air interface communications below 135 kHz", it defines the air interface for RFID devices operating below 135 kHz;
3. "Information technology—Radio frequency identification for item management—Part 3: Parameters for air interface communications at 13.56 MHz", it defines 3 MODES of operation, intended to address different applications;
4. "Information technology—Radio frequency identification for item management—Part 4: Parameters for air interface communications at 2.45 GHz", it defines the air interface for RFID devices operating in the 2.45 GHz Industrial, Scientific, and Medical (ISM) band;
5. "Information technology—Radio frequency identification for item management—Part 6: Parameters for air interface communications at 860–960 MHz", it defines the air interface for RFID devices operating in the 860–960 MHz;
6. "Information technology—Radio frequency identification for item management—Part 7: Parameters for active air interface communications at 433 MHz", it defines the air interface for RFID devices operating as an active tag in the 433 MHz band.

Apart from the above mentioned standards, there are other ISO/IEC standards that define other aspects of RFID systems, including the air-interface, operation of read, write, modify, and delete, data coding compression methods. Relevant such standards include "ISO/IEC 15961", "ISO/IEC 15962", "ISO/IEC 1596" and "ISO/IEC 24753". More detailed description about the above standards can be found from the Internet.

Considering that the identities of RFID tags must be different, the EPCglobal [1] developed industry-driven standards for the Electronic Product Code (EPC) to support RFID. The EPC code has four key attributes that enable the unique identification of an item: header, EPC manager number, object class, and serial number of the object. The size of EPC codes can be from 64 bits to 256 bits. A typical EPC code has 96 bits, this type of EPC codes have wide applications in supply chain due to their ability to generate a very large quantity of unique numbers. For example, for the 96-bit EPC code, 8 bits are used as the header, 28 bits are used to define the EPC manager, 24 bits are used to define the object class, and 36 bits are used to define the serial number. With this format, the EPC codes can provide unique identifiers for 268 million companies, each company can have 16 million object classes and 68 billion serial numbers in each class.

10.3 Security and Privacy Issues in RFID Systems

RFID tags are designed to identify goods and other kind of articles. While the most disadvantage of RFID tags compared with bar codes and QR codes is the cost. Nevertheless, RFID tags have many advantages that make the application of RFID systems widespread. RFID tags can communicate with a reader as long as they

are within the range of electromagnetic field of the reader, hence finding specific items from a pile of goods is more convenient with RFID tags. Multiple RFID tags can be read by a reader in a short period of time, making it more convenient than scanning each of the bar code. More importantly, RFID tags can be made to have some security functionalities, so that some attacks are difficult which would be easy with bar codes.

RFID systems face a number of security threats. If the RFID tags are devices holding some data, e.g., an RFID tag records the account balance for certain kind of consumption, then they face the threats that the data is stolen, or is illegally modified. In this case, data confidentiality protection is needed to avoid illegal readers to read data from the RFID tag, and reader/writer authentication is needed to avoid illegal writers to update the data, e.g., the account balance.

If an RFID tag is simply a device holding an identity, and all the associated information is stored in the data processing center, then the security threats are about the identity information of the RFID tag. The security threats on the identity information of the RFID tag is not simply the identity information as a data string, but the whole process how the RFID tag uses the identity information to respond a reader, which follows a standard protocol.

With respect to normal RFID tags, there are three main types of attacks: RFID tag cloning, RFID tag tracking, and relay attacks. These security threats and possible security solutions are further described.

10.3.1 RFID Tag Cloning Attack and Security Techniques

RFID tag cloning attack is to replicate a genuine tag. It is common that people replicate RFID tags, just like key cutting, and the convenience to clone RFID tags has obvious security vulnerabilities, because cloned tags may be used by attackers who can then lunch spoofing attacks. When RFID tags are used as electronic plates of vehicles, cloning of RFID tags can result in fake plates, which would cause a lot of problems in finding a vehicle that has escaped from a traffic accident.

It is easy to clone normal RFID tags. Given a genuine RFID tag, one can use a reader to get the identity information (e.g., the EPC number) of the RFID tag, and write the identity information into a blank RFID tag. The new RFID tag has the same identity information as the genuine one. It should be noted that, the device used to clone RFID tags may not be a normal reader, it is specially designed. It has the functionality as a reader that can communicate with an RFID tag to get its identity, record the identity information and use an RFID writer to write it into a blank RFID tag. The blank RFID tag must be of the same type as the genuine one being cloned. If the original genuine RFID tag is an access card, then the cloned tag can be used for the same access. This can be treated as an application of tag cloning, but has obvious security vulnerabilities.

In order to thwart RFID tag cloning attack, some methods are proposed, which are described as follows.

Use cryptographic method. With the technique of Electrically Erasable Programmable Read-Only Memory (EEPROM), and by using some cryptographic functions, RFID tags can be made to have resistance against cloning. A simple implementation is to use an encryption algorithm in both RFID tags and a reader, when an RFID is scanned by a reader, instead of responding with its identity, the RFID tag responds with the encrypted form of its identity, and the reader forwards the responding data to the data processing center which can decrypt the data to get the real identity of the RFID tag, hence can verify its authenticity. It would not be secure enough to write the encryption key in the EEPROM of the RFID tag, it is better to store the encryption key in an area that can hardly be accessed from external but can be accessed by internal logics, this property can be provided by security chips. Hence, by using a security chip, cloning the RFID tag is difficult. However, the cost of this approach is obvious due to special hardware and software for the RFID tags, and the corresponding process in the data processing center.

Apart from using an encryption algorithm to protect the RFID tags against cloning attacks, another method is to let an RFID tag authenticate the reader. By this approach, illegal readers cannot get the identities of the RFID tags, hence cannot clone them. However, this approach involves more sophisticated cryptographic techniques, and is more complicated than using an encryption algorithm.

Use a physical unclonable function. Apart from the cryptographic methods, some new methods have been proposed. One of such is Physical Unclonable Function (PUF) [5]. PUFs are a promising innovative primitive that can be used for authentication and secret key storage. Instead of storing secrets in digital memory, PUFs derive a secret from the physical characteristics of the integrated circuit (IC). For example, a PUF can use the innate manufacturing variability of gate delay as a physical characteristic to derive a secret.

10.3.2 RFID Tag Tracking Attack and Security Techniques

Large number of RFID tags are used in transportation and logistics, because the way to use RFID tags is much more convenient than using other visible codes. For example, EPC readers can read numerous passive transponders instantly, out to 10 m. This enables the identity scan while the items are in motion or inside a box. For a standard EPC tag, the communication from an RFID tag to a reader is simply to provide its identity information on request from the reader. Since the communication protocol between the reader and the tag is standard, any reader of the same type can get the same response from the tag, this characteristic results in security threats of tracking attack. The tracking attack is to use a reader to scan the identities of moving goods, and similar scan is attempted in different locations. When the scans from location A at time t_1 and location B at time t_2 get the same response (which does not have to be exactly the same, a fuzzy matching would be

OK), then it means that the goods with the RFID tag was in location A at time t_1 and in location B at time t_2. With more scans conducted, the moving goods can be tracked by the attackers, even though the goods may have changed the means of transportation.

If the RFID tags are used as name tags, then the tracking attack on a name tag can find the moving path of the person holding a specific name tag.

As described above, the identity of the RFID tags can be protected by an encryption algorithm. However, with respect to tracking attack, the following analysis shows that using different encryption algorithms has different problems.

Use deterministic encryption. A deterministic encryption algorithm transforms a plaintext into a fixed ciphertext under the same encryption key. This approach simply transforms the identity information of an RFID tag into its ciphertext format, it does not effectively provide protection against tracking attack, because the fixed ciphertext of the RFID identity can be used for tracking. The attackers do not care whether the response of the RFID tag is the plaintext identity, or the ciphertext of the identity, they only care about what the things are attached with the tag, as long as the tag responds the readers with the same responding data.

Use probabilistic encryption. A probabilistic encryption algorithm transforms a plaintext into a randomized ciphertext in the sense that every time the encryption algorithm is executed, the resulting ciphertext is different, and it is difficult to find the connection between different ciphertexts without the knowledge of the encryption key. There are different ways of constructing probabilistic encryption algorithms, and a preliminary realization is to employ a deterministic encryption algorithm but the message (the tag's identity) is attached with a random string. Using a probabilistic encryption can effectively combat the tracking attack, but the computation in the data processing center becomes costly, particularly when the number of RFID tags is large.

Assume that there are n RFID tags in an application. When a reader sends an encrypted RFID identity to the data processing center, the processing center does not know which is the decryption key, it has to try all the possible keys of the corresponding RFID tags and check whether the decryption matches the identity of the corresponding tag. On average, $\frac{n}{2}$ times of decryption algorithms (each with a different decryption key) have to be tried to find the correct matching, and in the worst case, it does n times of decryptions. If the communication between the reader and the data processing center can be forged, then a forged message with a random number to pretend the ciphertext of an RFID identity, it would cause the processing center to execute n times of decryption algorithm before the disturbance can be recognized, this makes the processing center vulnerable to DDoS attacks.

Use public key encryption. While the symmetric encryption has to use different keys for different RFID tags, with public key encryption, all the RFID tags can use a same public key encryption algorithm to encrypt their identities. Although many public key encryption algorithms are deterministic, they can be used to have probabilistic behavior by appending a random or pseudorandom number

to the tag's identity before encryption. The data processing center can simply decrypt the messages using its private key, to recover the identities of the RFID tag, regardless which tag encrypted the message. After decryption, by removing the random number, the identity of the RFID tag can be validated.
The disadvantage of this approach is that public key encryption algorithms usually require relatively large amount of hardware and software resources, and passive RFID tags may not possess the required resources.

RFID tag tracking attack is also known as RFID privacy attack, this is related to a location privacy problem when the RFID tag is used as a name-tag, and consequently the techniques to thwart the tracking attack are called RFID privacy protections [6, 8].

The above analysis shows that, it adds much cost to use an encryption algorithm to protect RFID tags from tracking attacks. However, there are other more efficient and practical methods to do this. Some of the effective techniques to combat RFID tag tracking attacks are described as follows.

Use the kill command. The EPCglobal standard specifies a functionality that can disables the functionality of the tag, this functionality is called the "kill command". The "kill command" is to make an RFID tag into the blank status as if it were just manufactured. After the "kill command", the tag cannot be tracked, because it has no connection to whatever it represented before. However, the major problem with the "kill command" is that, the command is performed manually, when it applies to large number of RFID tags, a small fraction of human error may result in large number of command failure, e.g., a careless execution of the "kill command" makes an RFID tag irreversible.

Use a hash chain. Each RFID tag contains circuits of two hash functions G and H. When the tag responses to a reader, the tag responses with $Res = G(ID)$, where ID is the identity of the RFID tag, and then updates its identity with $ID' = H(ID)$. The data processing center has a database recording each tag's original identity ID, its temporary identity ID', and expected response Res. When a response is valid, then the identity of the tag is validated, and the temporary identity is updated by $H(ID')$, and the expected response is also updated by $G(H(ID'))$ accordingly.
This scheme has good security, because given the Res, by the pre-image security of cryptographic hash functions, it is computationally infeasible to find what ID is. and the computational cost in the data processing center is a searching algorithm and two hash computation, which can be done efficiently. However, the scheme is fragile against synchronization attack, i.e., with a reading attempt by an illegal reader, the temporary identity of the RFID tag would update its identity with $ID' = H(ID)$, while the data processing center still records the identity as ID, this results in that the RFID tag is no longer recognizable by the data processing center.
If the RFID tags can validate readers, then illegal readers cannot change its temporary identity. However, implementation of identity authentication on RFID readers needs much hardware and software cost on the RFID tags.

Alternatively, the hash functions can be replaced by an encryption algorithm using a key shared by the RFID tag and the data processing center, after responding with the temporary identity ID, the RFID tag updates its identity ID with $ID' = E(k, ID)$, and the data processing center does the same computation. This approach can avoid the synchronizing problem caused by using two hash functions.

Use a challenge-response protocol. Challenge-response is an effective authentication mechanism, and it was used for RFID privacy protection by Molnar and Wagner [7]. Assume that an RFID tag and a reader (the reader also represents the data processing center) share a secret, then an authentication protocol can be established between the tag and the reader (which also represents the data processing center), which can avoid unauthorized readers to be able to read the identity information of the tag, hence can effectively combat the tracking attack. The authentication protocol can be depicted by Fig. 10.2.

The protocol in Fig. 10.2 assumes that a tag and a reader (actually the data processing center connected to the reader) share a secret k and a k-dependent one-way function f_k, which can be an encryption algorithm or a message authentication code algorithm. Then the process of the protocol can be described as follows:

1. The reader generates a random number α, and sends α to the tag;
2. The tag generates another random number β, computes $\sigma = ID \oplus f_k(0, \alpha, \beta)$, and sends (β, σ) to the reader, where ID is the identity information of the tag (can be treated as an integer);

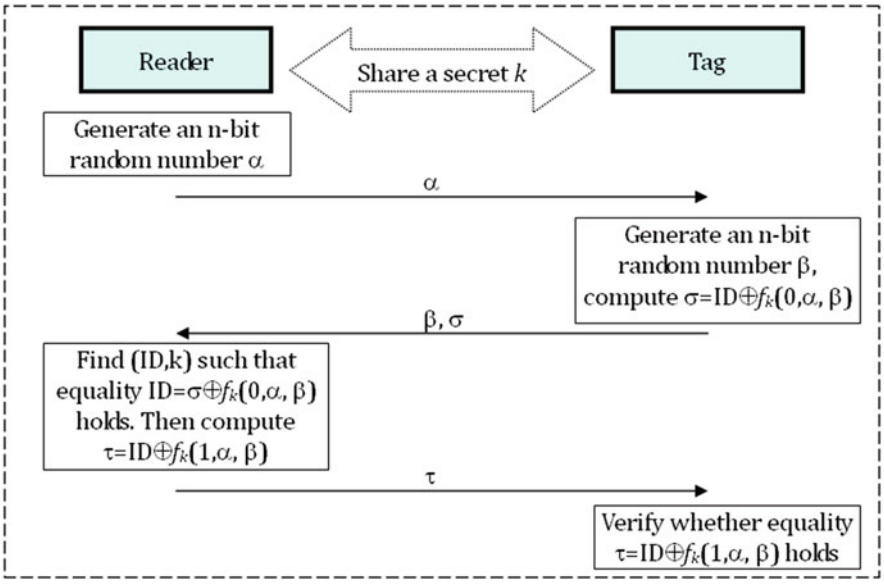

Fig. 10.2 An RFID authentication protocol

3. The reader goes through the tag identities recorded in the identity database, for each ID and its corresponding k, computes $\sigma \oplus f_k(0, \alpha, \beta)$ and check whether it equals ID. If equality holds, then that ID is the tag's identity. Then the reader computes $\tau = ID \oplus f_k(1, \alpha, \beta)$, and sends τ to the tag;
4. The tag computes $\tau \oplus f_k(1, \alpha, \beta)$ and check whether it equals ID. If so, then the reader is authentic and the authentication is successful.

The RFID authentication protocol described above uses 3-way communication, and much computation in the RFID tag is needed, hence it is not so lightweight. On the other hand, the protocol is inefficient, because the data processing center has to try each of the tag's ID and its corresponding secret key k, compute $\sigma \oplus f_k(0, \alpha, \beta)$ and verify whether it equals ID. On average, the computation has to repeat for $\frac{N}{2}$ times in order to find the correct match, where N is the total number of RFID tags. The efficiency decreases with the increase of the number of tags. In order to improve the efficiency, Molnar and Wagner proposed a tree-based scheme that can reduce the computational complexity from $O(N)$ to $O(\log N)$, and the cost is $O(\log N)$ storage at the tag.

There are many other RFID authentication protocols proposed in public literatures. However, the cost is that implementation of such authentication protocols requires extra hardware and software of RFID tags, and even needs to design private communication protocols between the RFID tags and the reader. The cost (financial or technical) might be intolerable for many systems.

10.3.3 RFID Relay Attack and Distance Bounding Protocols

When there is an authentication mechanism between an RFID tag and a reader, it seems difficult for unauthorized readers to get the identity information from genuine RFID tags, nor to spoof a genuine RFID tag. However, the attackers can make a rogue tag RT and a rogue reader RR. RT and RR can be physically far away from each other, but there is a communication connection between RT and RR, they send to each other the messages that they receive. This means that, whenever RT receives a message, it sends to RR, and when RR receives a message, it sends to RT. Assume that the attacker uses RR to scan a genuine RFID tag, and in the same time, the attacker puts the RT close to a genuine reader with which the genuine RFID tag is supposed to work, and RT sends a signal to the reader. When the reader sends an authentication challenge, RT receives it, sends it to RR which then sends to the genuine tag, and the tag sends a response back. When RR receives the response, it sends to RT which then sends to the reader as its response to the challenge. Since the response is made from a genuine tag, it is obviously correct. The whole challenge-response process between the genuine RFID tag and reader is hence completed by bridging communication channel simulated by RT and RR, and the authentication process can be successfully executed. This attack is called *relay attack*. The mechanism of relay attack can be depicted in Fig. 10.3.

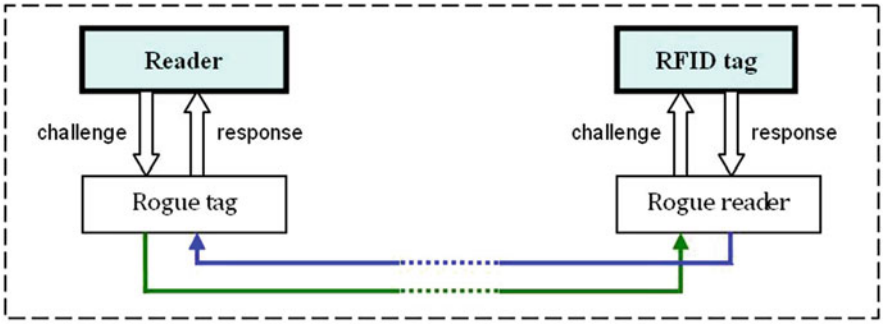

Fig. 10.3 The mechanism of relay attack

It can be seen that sophisticated techniques are required to lunch a relay attack. Although the relay attack involves high cost, if the reader controls an entry guard to a very important place, then the relay attack can be a big security threat for the possibility to cause unauthorized access.

While keeping the RFID tags in a safe place is a proper manner, however, human mistakes are unavoidable. In order to find technical solutions to thwart relay attack, it is noticed that, if the genuine tag is physically far away from the genuine reader, then the transmission of signal from the rogue reader to the rogue tag needs some time. Because electronic signals travel at the speed of light, the time of the signal transmission between the rogue tag and rogue reader (the same distance from the genuine tag and the genuine reader) is only a very short time, it is still measurable.

The technique of distance bounding [2, 3] was proposed based on calculating the round trip time (RTT) of the response after a challenge is sent. When the reader sends a challenge, the tag must respond with a response. This challenge-response process involves signal traveling a round trip. The reader can estimate the distance of a RFID tag to the reader by measuring the RTT given that the speed of the radio signal cannot exceed that of light.

A typical distance bounding protocol was proposed by Hancke and Kuhn in [4], which can be used to describe the mechanism of distance bounding technique. Hancke and Kuhn's distance bounding protocol assumes that the reader and the RFID tag share a secret k, and they both have a hash function H available.

Hancke and Kuhn's protocol consists of two phases: slow phase and fast phase. In the slow phase, the RFID tag and the reader exchange randomly generated numbers α and β. Then both the RFID tag and the reader compute a hash value $y = H(k, \alpha, \beta)$, which can be denoted by two n-bit halves, i.e., $y = y_0||y_1$. Here n is a security parameter. If the length of y is larger than $2n$, then y_0 and y_1 can be two parts of y that both the RFID tag and the reader agree. y_0 and y_1 are used as session secrets during the fast phase. In this sense, the slow phase is a preparation for the fast phase.

In the fast phase, for each $i = 1, 2, \ldots, n$, the reader sends a random challenge $c_i \in_R \{0, 1\}$ to the RFID tag, i.e., $c_i = 0$ or $c_i = 1$ is chosen at random. Then the reader replies with r_i, which is the i-th bit of y_0 if $c_i = 0$, or the i-th bit of

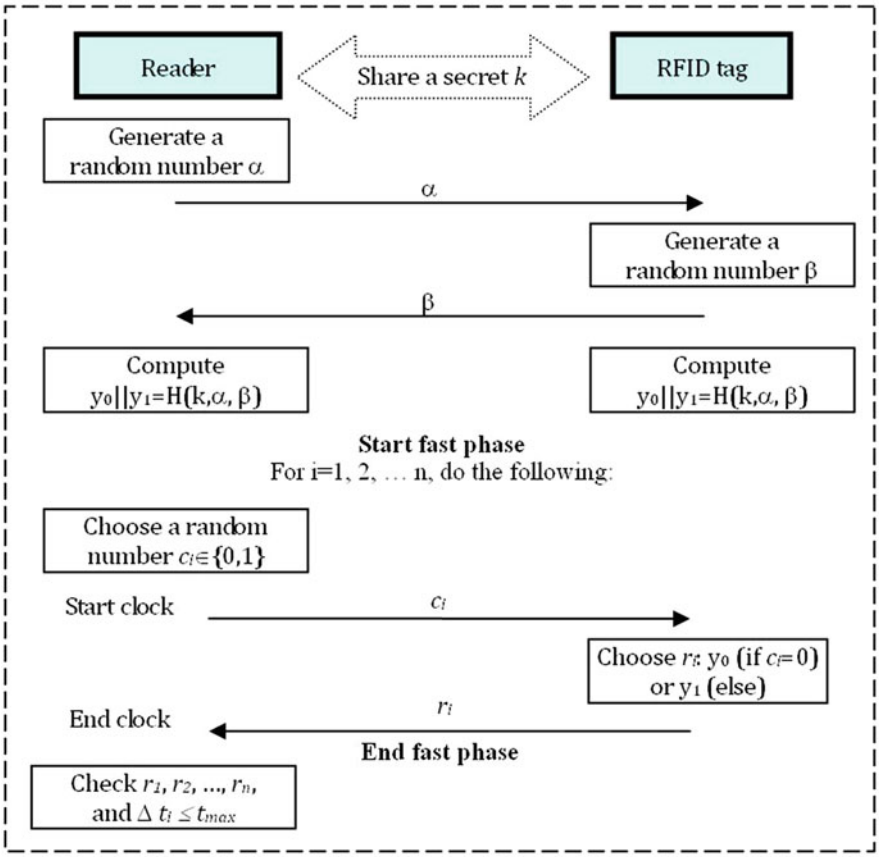

Fig. 10.4 Hancke and Kuhn's distance bounding protocol

y_1 if $c_i = 1$. For each rapid bit exchange, the reader checks the correctness of the received r_i and measures the round trip time Δt_i. After n rapid bit exchanges, the reader computes the number of rapid bit exchange where r_i is correct and $\Delta t_i \leq t_{max}$, where t_{max} is the maximum allowed time delay. If the correctness rate is acceptable, then the RFID tag is accepted. The process of the protocol can be depicted in Fig. 10.4.

Most of the distance bounding protocols use the RTT method to judge whether the tag is within reasonable distance. Assume that the reader sends a challenge at time is t_1, receives the response from the tag at time t_2, and assume that the time needed for the tag to compute (choose) the response is t_d. Then the maximum possible distance d_0 between the tag and the reader is

$$d_0 = \frac{t_2 - t_1 - t_d}{2} \times c,$$

where c is the speed of light. In a distance bounding protocol, it requires that $d_0 \leq d_{max}$, where d_{max} is the longest allowable distance, and it is obvious that $t_{max} = d_{max}/c$.

Since the response to each of the challenge c_i is 0 or 1, a random guess can be correct with probability 50%. The probability for all the n guesses to be correct is $\frac{1}{2^n}$ which decreases exponentially with the increase of n. This means that n does not have to be very large when $\frac{1}{2^n}$ becomes negligible.

Instead of random guessing, the attacker can make use of a genuine RFID tag. Once the tag and the genuine reader finished the slow phase, if time slot for the slow phase is relatively long, the attacker can simulate a sequence of queries c_i to the genuine RFID tag to get n responses r_i, and use them to respond the challenges from the genuine reader, where c_i is randomly generated, because there is no way better than random guess. For each challenge c_i' made by the genuine reader, it is obvious that the probability for $c_i = c_i'$ to hold by coincidence is 50%, in which case the response r_i is definitely correct; The other 50% of chances are that $c_i \neq c_i'$, in which case, the probability for r_i to be coincidentally correct is 50%. Hence, the overall probability for the attack to make a correct response in each rapid challenge-response is $0.5 + 0.5 * 0.5 = 0.75$, hence the probability for all the responses to be correct in n rounds of rapid challenge-response is 0.75^n, which decreases very fast with the increase of n.

If the longest allowable distance d_{max} is restricted to be small, e.g., tens of meters, then when the distance between RT and RR is longer than 100 m, if the response from RT uses the message from RR, then the waiting time will exceed t_{max}; on the other hand, if RR uses a kind of random guess to respond the fast phase, then the chances to respond correctly is no more than 0.75. Hence, the number of rapid bit exchange satisfying that the response r_i is correct and $\Delta t_i \leq t_{max}$ is likely to be smaller than $0.75n$, where n is the total number of exchanged bits. This means that relay attack can be detected.

It can be seen from Fig. 10.4 that, the distance bounding protocol is a special protocol, and it needs a pre-agreed secret key between the tag and the reader, a hash function and some storage, and the reader has the capability of clock measure with high precision.

The distance bounding protocols in theory can thwart RFID relay attack. However, when the genuine tag is close to the genuine reader (e.g., the company's car park), and the devices used for relay attack has good computation and communication capability that consumes very little time for message relay, then the distance bounding protocols may fail to work. It is obvious that the cost of relay attack is high, while it can be prevented by a good management policy. For example, if an RFID tag is put into an electromagnetic shielding bag when the tag is not being used, then this simple physical operation can effectively defeat the relay attack.

10.4 Summary

RFID systems have wide applications in IoT systems, such as smart logistics, smart vehicles, source-tracking, and name tags. An RFID system consists of RFID tags, one or more RFID readers/writers, and a data processing center. The communication between an RFID reader to the data processing center uses a traditional network connection, which can be over the Internet or using mobile communication systems, hence the security is not a big concern in RFID systems. The communication between an RFID tag and a reader/writer uses wireless connection and usually standard protocols, this is where security problems are concerned.

The original purpose of RFID tags is to record identities. This makes RFID tag cloning possible. In fact, tag cloning happens in real life, which can make convenience, just like key cutting. However, the availability of cheap RFID readers and writers makes RFID tag cloning a security threat, because unauthorized users may have chances to clone genuine RFID tags and gain access to areas where their access is not authorized.

Apart from RFID tag cloning, other security threats include RFID tag tracking and relay attacks. RFID tag tracking is a security threat to moving items. If the tags are used as name-tags for company employees, then the tracking attack can find the dynamic location of the employees, hence the tracking attack can cause RFID privacy problems. If the tag cloning problem can be solved, then relay attack still works. The relay attack is to forward the communication between a genuine tag and a genuine reader by a rogue reader and a rogue tag. Since the relay attack works like the communication occurs between a genuine tag and a genuine reader, even though there is an authentication protocol between the tag and the reader, the authentication protocol can be executed successfully. If the relay attack is used on an access card, then a successful relay attack can gain unauthorized access.

Distance bounding protocols are designed to thwart the relay attacks. However, the distance bounding protocols have a relatively high requirement on hardware and software, while practically the relay attack can be prevented by little effort in tag management, e.g., putting an RFID tag into an electromagnetic shielding bag. Hence, the techniques described in this chapter are of more academic significance than practical applications.

It should be noted that, apart from the security threats described in this chapter, there are other kinds of attacks on RFID systems, such as side-channel attack and physical dissection attack. Such attacks involve sophisticated laboratory equipment. However, if large amount of RFID tags are used in a critical application, and they share a common secret key, then attackers would be willing to perform side-channel attack and even physical dissection attack.

With the development of IoT applications, RFID tags are used in more and more applications. Hence, new kinds of security threats may emerge, and some of the previously not-so-practical attacks may become effective attacks in the real world.

References

1. J.I. Aguirre, EPCglobal: a universal standard. Master Thesis (MA 02142), MIT, working paper CISL#2007-01, Feb 2007. http://hdl.handle.net/1721.1/42350
2. T. Beth, Y. Desmedt, Identification tokens - or: solving the chess grandmaster problem, in *Advances in Cryptography - CRYPTO 1990*. Lecture Notes in Computer Science, vol. 537 (Springer, Berlin, 1991), pp. 169–177
3. S. Brands, D. Chaum, Distance-bounding protocols, in *Advances in Cryptology – EUROCRYPT 1993*. Lecture Notes in Computer Science, vol. 765 (Springer, Berlin, 1994), pp. 344–359
4. G.P. Hancke, M.G. Kuhn, An RFID distance bounding protocol, in *Proceedings of International Conference on Security and Privacy for Emerging Areas in Communication Networks, Athens* (IEEE Computer Society Press, Washington, 2005), pp. 67–73. https://doi.org/10.1109/SECURECOMM.2005.56
5. C. Herder, M. Yu, F. Koushanfar, S. Devadas, Physical unclonable functions and applications: a tutorial. Proc. IEEE **102**(8), 1126–1141 (2014)
6. M. Ohkubo, K. Suzuki, S. Kinoshita, RFID privacy issues and technical challenges. Commun. ACM **48**(9), 66–71 (2005)
7. D. Molnar, D. Wagner, Privacy and security in library RFID: issues, practices, and architectures, in *Proceedings of the 11th ACM conference on Computer and communications security (CCS 2004)*, Oct 2004, pp. 210–219
8. S.E. Sharma, S.A. Wang, D.W. Engels, RFID systems and security and privacy implications, in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*. Lecture Notes in Computer Science, vol. 2523 (Springer, Berlin, 2003), pp. 454–469

Chapter 11
On the IT Security and the OT Security in IoT

IT security is the abbreviation of information (technology) security. There are many mature techniques in IT security protection. OT security is the abbreviation of operation (technology) security, which is relatively a new concept and lacks relevant techniques. This chapter clarifies the concepts and features of IT security and OT security, discusses their differences and how they are related to IoT applications. Different security goals and security protection mechanisms are discussed with respect to IT security and OT security respectively, where a security architecture of "intrusion tolerance" is introduced targeting at resisting OT attacks. The proposed intrusion tolerance is an architecture suitable for IoT devices, and has substantial differences from the existing intrusion tolerance techniques proposed for IT systems.

11.1 Introduction

From the three-layer IoT architecture (as shown in Fig. 2.1) it can be seen that, the perception layer of IoT includes sensors and controllers/actuators as common IoT devices. Sensors collect environmental data and transmit to a data processing center, and the controllers/actuators control terminal devices under the receipt of command instructions from a data processing center. This simple data transmission process goes through three layers, the perception layer where the sensors and the controllers are, the network layer which works as a tunnel transmitting the sensor data and the command data, and the processing layer where the data processing center resides.

When the data from perception layer is transmitted to the processing layer, it may encounter malicious attacks. Traditional techniques for data security protection are targeted at protecting data confidentiality, data integrity, data source authentication, etc. In IoT applications, there are some special requirements on the security services. For example, in traditional Internet environment, data freshness is not an important security measure, because usually data is split into small IP packets while they are

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_11

transmitted over the Internet, different IP packets may go through different paths, so that the data packets will be assembled together when they arrive the destination. Sometimes more than one copy of an IP packet may arrive the destination. Because each IP packet has a unique IP packet ID, the replicate copy will be ignored. In this sense, replay attack on certain IP data packets do not work, while replay attack on the whole data is difficult to lunch, unless attackers reside very near to the data source or data destination. However, in IoT applications, since the size of application data can be very small, such as a command, one IP packet is often sufficient to hold the whole piece of data, so replay attack on one IP packet can be a replay attack on the whole command data, which may become a serious security threat. Hence, data freshness as a security service to thwart replay attack becomes an important security measure in IoT applications.

On the other hand, many low-cost IoT devices have limited capability in computation and communication, particularly those wireless IoT devices that use a battery to supply power, most traditional security techniques are not suitable to those IoT devices, they need security techniques with the feature of being lightweight. Compared with traditional security techniques, the corresponding lightweight security techniques require smaller hardware space (measured by the number of GE—gate equivalence), less codes in software implementation, and consume relatively less power. For this sake, many lightweight encryption algorithms have been designed [5, 14].

Among those lightweight security techniques, lightweight authentication protocols are more important, because authentication protocols consume energy in computation and communication, while a lightweight encryption algorithm only consumes energy in computation. In terms of energy consumption, our experiments show that data communication consumes much more energy than cryptographic computation on the same data. The idea to reduce data communication cost is to minimize the number of data transmission rounds. For example, traditional identity authentication protocols often use the technique of challenge-response, where challenge and response need at least two rounds of data communication. Often an authentication protocol provides confirmation to tell the prover (the one that does the response) that the protocol is successful, or otherwise an error message, this requires one more round of data communication. In IoT applications, in order to minimize communication cost, confirmation in an authentication protocol is not necessary, even the technique of challenge-response is used, two rounds of communication would be sufficient to complete an authentication protocol. With some dedicated design, one round of communication would be able to complete an identity authentication with practical security.

While information security techniques are needed in IoT applications as in traditional information systems, an IoT system may also need techniques for operation security, i.e., the security safeguarding operation and control of IoT devices. In order to differentiate operation security with traditional information security, this book treats traditional information security as *IT security*, and operation security as *OT security*.

11.2 The IT Security and OT Security in the PERA Model

Security protection becomes necessary for most of the information systems. It is known that, traditional security techniques provide services including message confidentiality, data integrity, and availability.

The confidentiality is a security service ensuring the data content cannot be accessed by unauthorized users, which can be achieved by using an encryption algorithm. The integrity is a security service ensuring that the data format has not been illegally modified, which can be achieved by using a message authentication code (MAC) algorithm, and the availability is a security service which ensures the system works properly for authorized users. Techniques to achieve the availability service include those against DoS and DDoS attacks, those of authentication and authorization, and perhaps those of access control.

With the interaction between information systems and physical operations, the OT technology has been an important component in IoT systems. Correspondingly, there should be security techniques for the OT technology, since the security techniques for IT systems may not be suitable for safeguarding OT related activities.

Apart from IT security, IoT systems also need OT security services. The target of OT security is mainly to protect illegal control and operation, which is particularly important in industrial Internet of Things systems.

Based on the Purdue Enterprise Reference Architecture (PERA) model, an industrial control system (ICS) network has six levels, they are as follows:

1. Level-0 is about physical process;
2. Level-1 is about basic control, e.g., sensing and manipulating the physical processes;
3. Level-2 is about area supervisory control;
4. Level-3 is about site operations, managing production work flow to produce the desired products;
5. Level-4 is about site business planning and logistics, managing the business-related activities of the manufacturing operation;
6. Level-5 is about enterprise network, analyzing the overall production status, inventory, and demand.

It can be seen that, Level-0 to Level-3 are within the manufacturing zone, exchanging data with Level-4. Operation technology is largely used in Level-1 and Level-2, while Level-3 is mainly used as an interface for human monitoring and on-site control.

Level-4 is a typical IT system that supports the production process, making the production process better meeting market demand (e.g., orders of customers). In an industrial IoT system, Level-4 can often be accessed from the Internet.

Level-5 is the enterprise network, which is likely connected to the Internet. Sometimes Level-5 is treated as a sub-level of Level-4, because Level-4 and Level-5 are both like traditional information systems.

Although the operational technology is employed below Level-4, the OT security techniques are likely allocated and executed in Level-4, which is where the OT security attacks are made. Currently, the security techniques applied to Level-4 and Level-5 are mostly traditional IT security techniques, because they are indeed IT systems. However, since Level-4 connects with Level-3, Level-2, and even down to Level-1, the operation behavior can be controlled in Level-4, hence OT security and IT security are both needed.

11.3 IT Security Versus OT Security

There may be confusion about the differences between the security techniques of IoT systems and those of traditional information systems. The security techniques are corresponding to the security requirements. In traditional information systems, basic security requirements include confidentiality, integrity and availability. The confidentiality is a security measure against information stealing/leaking, which can be achieved by using an encryption algorithm. The integrity is a security measure against data modification attack, which can be achieved by using a message authentication code (MAC) algorithm, and the availability is a compound security measure which can be achieved by a number of techniques and management processes.

Intuitively, the IT security of IoT systems is the same as that in traditional information systems. This is true from the respect of security functionality. However, from the respect of practical implementation, there are some substantial differences. Below are some of the differences.

1. Most endpoint IoT devices have special security requirements. For example, upstream data from sensors to a data processing center often need data confidentiality service, because upstream data are usually application data, while the downstream data from the management center (sometimes from end user via the management center) often need data integrity service, because the downstream data usually command data.
2. Many IoT devices are resource-constrained. Even though the devices themselves may be of low cost, and the data they proceed may be valuable, hence security protection is necessary. There should be a lightweight feature of the security techniques used in many IoT systems with resource-constrained devices. Many lightweight cryptographic algorithms [1, 17, 18] hence have been designed.

Apart from IT security, the requirement of OT security protection makes IoT security much different from that of traditional IT systems.

11.4 Characteristics of Network Attacks Targeting at OT Security

Compared with the IT security, the OT security in IoT systems is very different. This is particularly the case in industrial control systems (or industrial Internet of Things, IIoT). When an attacker successfully intruded into an IIoT system, for example a master control station (MCS), possible attacks made by the attacker include the following types:

1. The first type of attacks are made by viruses that are designed to damage or even to destroy the infected systems. Computer viruses[1] in the early age were targeting at damaging the computer systems wherever they can infect. There can be different ways to damage the computer systems, for example, to exhaust the computers' memory/storage/network resources, or to damage/destroy some existing data. Such damage is mainly associated to software and data. Since the destruction attack does not seem to bring attackers any reward, this kind of attack becomes very rare now.
2. The second type of viruses are designed to gain illegal profits. One way to get profits by the viruses is to collect some valuable data from the infected systems, and get benefit from selling such data. Another way is to kidnap the infected systems, forcing the system users or data owners to pay some money to hopefully get the kidnapped systems recovered. The ransomware appeared in recent years such as Wannacry [15] is of this kind.
3. The third type of viruses are designed to illegally control a physical system, and the Stuxnet [7] is a typical such example. Such attacks may be for financial or political purposes. This type of viruses is often designed targeting at a specific industrial control system, where the designers have good knowledge about the hardware, operating system, and industrial configuration software of the target system. The attack is done by first to infect a master control station and try to illegally control the devices/machines that are connected to the master control station. rather than to collect information (although some viruses really do so) or to do some damage on the infected systems. In order to increase the chances of successful attacks, such kind of viruses would try to hide themselves as much as possible, The purpose of this type of attacks is to damage the endpoint physical devices/machines connected to the infected computer systems, hence to damage the whole industrial control system.

The first two types of attacks are targeted at traditional information systems. The third type of attacks are usually designed for specific industrial control systems or IIoT systems, where the intruder (the virus) has high skill of hiding itself, and is much more sophisticatedly designed.

[1]The term of "virus" in this chapter is a general term, which may mean any kind of malicious code.

Damaging the infected systems or to collect/steal selected data from the intruded systems is the behavior of attacks on traditional information systems. For industrial control systems or IIoT systems, the damage on a control computer (e.g., an MCS) causes limited damage to the whole IIoT system, since the control computer can be replaced by a backup, or the control work can be taken by another computer. For an IIoT system, the most severe attacks are targeted at forging control operation, such kind of attacks are called operation technology based attacks (or *OT attacks* for short), and the security problems related to OT attacks are called OT security problems.

Although OT attacks do not target at collecting information from the intruded systems, practical attacks are often complicated, including both attacks on information systems (called information technology based attacks, or *IT attacks*) and OT attacks.

For an IoT device, denote by δ ($0 \leq \delta \leq 1$) the level of IT attack and by ω ($0 \leq \omega \leq 1$) the level of OT attack, where $\delta = 0$ means that the system has no IT attack at all, $\delta = 1$ means that the system is completely down by the IT attack. Similarly, $\omega = 0$ means that no malicious and illegal control on other IoT devices has been made successful, and $\omega = 1$ means that all the illegal control commands on other IoT devices are successfully executed. Then the security status of an IoT system can be measured by the pair (δ, ω). In the case of IT attack, there is no attempt to control other connected devices, hence $\omega = 0$; In the case of OT attack, there is no attempt to affect the current device, hence theoretically we would have $\delta = 0$. However, the activity of OT attack (regardless whether illegal control attempts are successful) consumes some resource of the current device, and perhaps with some illegal operation, hence the value of δ is practically not zero. In order to cause less effect on the current device when lunching an OT attack, attackers try to make δ as small as possible, unless IT attack is part of the designed purpose of the attackers.

11.5 Characteristics of OT Security

The term of IT security is a general concept of security techniques for information systems, which is supported by information technology. The term of OT security is closely related to IT security, which is a general concept of security techniques for operational activities, which is based on operation technology. IT security is mainly for information systems, while OT security is mainly for the operation and control of physical devices. In industrial control systems, OT security is the core security, because safeguarding an industrial control system to have stable operation is the most import goal of industrial control system security services [6].

Today, the well-developed network infrastructure provides strong support to the application of Internet of things, including the networking of industrial control systems. Such networked industrial control systems can be developed to Industrial Internet of Things (IIoT). However, while the IIoT provides many advantages

over traditional industrial control systems, such as convenient operation and management, better control of the product quality, better management of production-storage-sale chain, and better meeting customers' demand, there are also many new security threats coming from the network environment. It is unfortunate that, if an industrial control system is physically isolated from the Internet, there is still a chance for the Internet warms to intrude the industrial control system. Therefore, security protection is necessary not only to the IIoT, it is necessary to isolated industrial control systems as well.

It is known that IT security includes confidentiality (for data content protection), integrity (for data format protection), and availability. In order to differentiate the OT security with IT security, by analyzing the features of security threats to the industrial control systems, the features of OT security can be described as follows.

1. **System stability.** When a control command is executed by a remote device, the control command and execution feedback need to be transferred over the network (often via wireless networks). The success of such data exchange is subject to many factors, including the reliability of the network (such as network connectivity and redundant connection, signal redundancy, device backup), timeliness of communication (ensuring data transfer can be completed within an allowable time slot), interference resistance (enabling normal data transmission even in the case of occasional signal interference.
2. **System robustness.** If the control commands change at a great scale during their sequential execution, the control system should have appropriate policies to execute the commands, while the effect of the system made by the big change of the commands should be minimized. How to set such policies is subject to the characteristics of the system. For example, if a system usually works smoothly, when a command instructs the system to make a sharp change, the smoothness of the system will be affected, in this case, gradual execution of the abruptly changed command would be a good policy. However, if abrupt changing of commands is a normal phenomena, then a good policy would be to execute the new commands as soon as possible.
3. **System controllability.** A user of an industrial control system may have very limited right to manage the data and processes of the system. If the system needs to fix a problem, or even needs regular maintenance, it is often done by the manufacture or the system constructor/operator, regardless whether the user trusts the manufacture or the system constructor/operator. In order to reduce the cost on traveling, many industrial control systems are connected to the public networks, allowing technical people to conduct the maintenance or problem solving remotely over the network. Although access to the network needs authentication, it still leaves a big security threat to the system. If an industrial control system can be operated and maintained by the users themselves, then the system is treated as being controllable by the users, because in such case, access by the manufactures and system operators via the network can be disabled. Whether a system is controllable by the users themselves is called the controllability problem.

4. **The legibility and authenticity of the control command.** The legibility of a control command is usually defined by the specific system. For example, if a command is beyond the capability of practical operation, then execution of such a command may cause damage to the system. Many application systems have rules to eliminate the execution of such commands, i.e., have capability to check the legibility of control commands. Even if a control command looks eligible, there is still a problem of the authenticity of the actual source of the command. Although the authenticity of the control command looks similar to the authenticity problem in other information systems, in IoT applications, the techniques used to provide the authenticity service can be very different.

The differences between OT security and IT security are essential. In industrial control systems [2], the OT security is more concerned than the IT security. A typical example is the security problem in smart grid in IIoT environment [3], this is because the system intrusion in an IIoT system may not threat the normal function of the system, provided that illegal and malicious control is not successful. In an industrial control system, or an IIoT system, some master control computers may have been continuously working for many years, with out-of-date configuration and no patches upgraded. Such systems have serious vulnerability against network intrusion, even if such systems are physically separated from the Internet. In fact, many master computers in industrial control systems have already been affected by malware, since those malware are not targeted at industrial control systems but traditional information systems, aiming at collecting data from the intruded machines, hence they do not cause operational damage to industrial systems. In this sense, those master computers in industrial control systems have tolerance against IT intrusion.

11.6 The Limitations of IoT Attackers

Before any intrusion takes place, the very first step to do an intrusion is to find a target. Apart from some important industrial control systems, where attackers can get the information of such systems from public media and other means, for normal IoT devices, the best way to find a target to attack is by network searching. Once an IoT device is found to be connected to a network, particularly when it is connected to the Internet, the attacker can follow a number of steps to intrude into the IoT device, hence can try to control it, if there is a security flaw to enable the intrusion.

Given that, the first security policy is to protect the IoT devices from intrusion. Unfortunately, regardless how the IoT devices are protected against intrusion, there is still a possibility of being intruded. Security protection can increase the difficulty of intrusion, and can hardly avoid intrusion. Many IoT devices have constrained resources, which would limit the level of security protection [9], For the IoT devices that are not yet intruded in certain period of time, intrusion may occur in the future. For low cost IoT devices, it is difficult to implement security protection against

intrusion, and intrusion targeting at OT attack may not cause serious damage, if illegal operation is not successful. For such IoT devices, the OT security is usually more important than IT security.

Once an attacker intrudes into an IoT device, the attacker may take some illegal action. Note that the device may not be an IoT endpoint device, for example, it may be an IoT gate node that connects with a number of IoT endpoint devices via local area networks. In this case, the attack is said to be successful if it can successfully control some IoT endpoint devices via the gate node. For the purpose of such kind of attacks, the attacker needs to have sufficient knowledge about the intruded IoT device and its connected IoT endpoint devices. This scenario is where OT security makes specific meaning, and makes the essential difference from IT security.

Even if an intruder successfully accesses into an IoT endpoint device, it does not mean that the intruder has complete control over the device. There are a number of factors that affect how much control one can do. For example, accessing into an device with different role may have different right of control over the device. Commands over the networks may need to be equipped with security services such as identity (e.g., source address) authentication, command data integrity protection, command data validity verification, and data freshness verification. Hence, sometimes the knowledge of the security parameters (e.g., a valid symmetric encryption key, a valid private key for signature) determine whether an OT attack can be successful.

If the intruded IoT device is an IoT gate node, where the gate node is a master control computer in an industrial control system (or an IIoT system), then whether the intruder can lunch a successful attack is subject to many factors including the following:

1. The control commands may have security protection, such as being encrypted, having integrity protection with a message authentication code, and other security verification with additional data;
2. Even if control commands have security protection, the intruder may not be able to execute the same security protection functionalities. One of the special cases is when an external hardware (e.g. U-key) is used to perform the command data encryption, in such case, the intruder cannot conduct the correct encryption of the command data, and hence cannot succeed in doing an illegal control operation.
3. The intrusion may be detected and removed before it causes any damage. Some IoT devices have certain level of intrusion detection. When an intrusion can be detected, it may be removed. This means that, a successful intrusion does not necessarily yield a successful attack;
4. The attack may not cause obvious effect to the working status of the intruded device. For example, an IoT gate node with intrusion tolerance may be intruded, but the intrusion causes no damage, and the IoT gate node can keep working as usual.

It is known that many IoT devices have constrained resources, and can only have very basic security protection. The reality is even worse, many IoT devices do not have security mechanism at all. The damages caused by IoT device intrusion can be

very serious, especially in IIoT [11]. Collective behaviors of IoT devices [12] may be useful to help to detected whether some IoT devices behave abnormally, however the resource-constraint feature of many IoT devices may eliminate the capability of collaboration among IoT devices.

11.7 Other Security Measures Related to OT Security in IoT Systems

In an IoT system, the OT security has close connection with the IT security. The data used to instruct an operation supported by some OT techniques may be processed using IT techniques, while the results of operation using OT techniques may need to be further processed using IT techniques. Some IT security problems may affect the OT security directly, and OT security problems may result in IT security problems. For example, abnormal behavior of IoT devices caused by OT security problem may result in abnormal data, which is seen as an IT security problem.

There are common techniques that can be used both in IT security and in OT security. For example, data encryption and data integrity protection can be used in both OT security and IT security. However there are some specific OT security techniques for operation technology, which have obvious differences from traditional IT security techniques for information systems. Such security techniques include the following: system stability, robustness, and controllability.

Note that the main service provided by OT security is to ensure that the control command data is correct. With respect to the content of the command data, the most intuitive way of operation of such data is to check whether the command data matches any of the pre-defined commands, if so, then the data as a command is executed, otherwise, the command data is discarded. This process is like an "if-else" logic.

When the control commands are represented as communication data, security protection for control command data also includes confidentiality and integrity protection. In many cases, the content protection of the control command is not necessary, since it may be easy to guess the content of the command. For example, a switch operation command has two values representing "on" and "off", hence its confidentiality makes little sense. However the integrity protection is extremely important, any illegal altering of the control command may cause unexpected damage. In addition to the integrity service of command data, the source authenticity of the command data also needs to be verified. Fortunately, since the command data is normally in small size, it is possible to integrate the source authenticity and the integrity of the command data together, yielding the protocol to be lightweight. A more sophisticated protocol may be able to integrate a number of security services together, i.e., including data source authentication, data integrity protection, and data freshness verification.

It should be pointed out here that, when command data needs content protection service, i.e., to protect its content from illegal users, a probabilistic encryption algorithm is recommended. With a probabilistic encryption algorithm, viewers on the ciphertexts can hardly guess the corresponding content, it is even computationally infeasible to identify whether two ciphertexts correspond to a same plaintext.

Give the special features of OT security techniques, some specific OT security related requirements are further analyzed below.

11.7.1 Stability of IoT Systems

The stability of IoT systems is a security measure, it is related to the availability service (an IT security measure) and proper operation (an OT security measure). The stability of IoT systems is related to proper transmission and execution of command data. It is an important security measure in industrial control systems and IIoT systems, since reliable working status is the most important thing for an industrial control system, regardless whether or not the system has intruders inside.

The stability of industrial control systems is reflected from the process control stability. In an industrial control system, control is a process of continues adjustment based on the system information feedback and some external information. Although a practical control system is quite complicated, a simplified theoretical model is often taken as a linear control system [2], which can be described by the following equation:

$$\bar{x}(t) = A.x(t) + B.u(t)(t > 0),$$

where $x(t)$ is the state vector, $u(t)$ is the control input signal, they both change with time t, A and B are constants.

The concept of stability is closely related to the availability of information systems, which is one of the three most significant security measures. The system availability is usually regarded as a security measure in industrial control systems, hence it is also an important security measure in IoT.

11.7.2 The Robustness of IoT Systems

The robustness of IoT systems measures the capability of partial failure tolerance, it is another OT security requirement. When some parts of an IoT system do not work properly, while the whole IoT system still work and can fulfill the basic task, then the IoT systems are said to have robustness. The level of robustness can be different, it is measured by how much failure an IoT system can tolerant.

If an IoT system is properly designed to have a good robustness and is so constructed, then the system can have much higher reliability than its parts, this

means that the failure of parts of IoT devices does not cause the system failure. Common techniques to make a system robust include the following: redundant setting of IoT devices, redundant network connections, redundant computation, and redundant power supply.

Practically, system redundancy is designed system-wise instead of device-wise, targeting at improving the robustness of the whole system. There are large amount of research about network redundancy. For example, the whole lifetime of an IoT system can be extended by balancing the energy consumption by each IoT node [4], or to reduce the running cost of the network when its lifetime has been fixed [10]. Sometimes the system robustness requires to ensure the correct result of the perception data even though small account of sensors do not work properly [16], which means that the robustness and stability are closely related.

Obviously the price for providing system redundancy involves extra cost, both in system construction and system operation. For example, if an IoT system having n IoT devices provides a redundant backup for every device, then the whole system needs $2n$ IoT devices, which doubles the cost of the system's hardware and software. Practical implementation can make further optimization, for example, same type of devices do not deed to a backup for each of them, a relatively small number of backups are sufficient, the number of backup devices can be determined based on the chances of device failure, including the failure caused by natural hardware or software fault and network attacks.

11.7.3 The Controllability of IoT Systems

In practical applications, the controllability of IoT systems (especially the IIoT systems) is treated as a key factor of IoT practical security, it is the level of confidence on IoT security from the respects of the system design, implementation, operation, management, and supervision.

The controllability of IoT systems includes the following aspects:

1. The controllability of hardware, i.e., the reliability of the hardware used in IoT systems is believed to have no hidden operations (e.g., a hidden backdoor) exist.
2. The controllability of software, i.e., the mechanisms and functionalities of software (including the operating systems) are believed to have no hidden malicious operations. Although most of large software exist security flaws, the controllability only ensures that no backdoors and security threats are embedded by purpose.
3. The controllability of system operation, i.e., during the system operation (including the execution of software and operation of hardware), no unauthorized processes, data, operations are accepted. This is to ensure that the system operation behaves in compliance with a rational expectation.

4. The controllability of the whole IoT system, both technically and by management/supervision, in the processes of system construction, system operation and maintenance, and system application.

From management aspect, the controllability can be understood as to include the following:

1. to minimize the chances of system fault caused by natural accidents and misoperations;
2. to minimize the chances of malicious intrusion, attack, and destruction, and to increase the capability of finding such malicious activities in time;
3. to minimize the restriction of intellectual property protection, for example, to use techniques based on self-owned intellectual protection;
4. with respect to malicious activities, apart from technical protection, it would be a good complement to seek legal assistance based on technical evidence.

From technical aspect, the controllability can be understood as to include hardware, software, and operating systems. With respect to hardware controllability, a closely related technique is called trust computing. In 1999, a trusted computing platform alliance (TCPA) was established by the collaboration of Intel, IBM, HP, Microsoft, and other companies. The TCPA defines a trusted platform module (TPM), which is meant to provide system security starting from hardware. By embedding a security chip on the motherboard, which in cooperates with higher layer software and trusted software stack (TSS), the TPM can filter unauthorized software and processes from being installed and executed, aiming at fundamentally eliminating security attacks. In 2003, the TCPA organized a more commercialized trusted computing group (TCG) with new industry members such as Sony and Sun.

Although the TPM defines a trusted computing environment, the meaning of trust here is in the sense of hardware devices. Inspired by the idea of TPM, China defined a trusted cryptography module (TCM) utilizing some Chinese national standard cryptographic algorithms. Security chips with TCM architecture can be used to provide security services for system platforms and application software.

With respect to the controllability of software, it is mostly about the purpose and implementation of the software. In the process of software production, if the software is made for the purpose of the user's requirement, and the software maker and the users belong to a same interest group, then the software user can be treated as having good controllability over the software. In the process of software application, the controllability is mainly about the functionalities and access boundaries, i.e., a software is supposed to do whatever it is designed to, and nothing else.

11.8 Compliance Verification of Commands in IoT Systems

The OT security of an IoT system is mainly about the transmission and execution of command data. The authenticity of command data can be ensured by an authentication protocol. However, an authorized command may not be legitimate, i.e., it is not compliant with the expectation of a legitimate command. For example, the

command may be illegitimate due to occasional human mistakes, or the command is purely a forged one aiming to cause damages. The compliance verification of a command is to check whether the command complies with a number of pre-defined rules. Below are a few examples for checking whether a command is legitimate:

1. The format of the command is correct;
2. The value of the command is within a reasonable pre-defined range;
3. The command arrives at the right time, e.g., when the IoT device is ready to accept and to execute new commands.

The above rules can be implemented by simple computation such as string matching. Of course there are more sophisticated compliance verification techniques. For example, based on the curve of the historical command changing, it is possible to determine whether a new command is legitimate. Usually such a conclusion probabilistic, i.e., how likely a new command is judged as legitimate or illegitimate. It seems that some artificial intelligence techniques may be useful to the sophisticated compliance verification. To avoid possible confusion, below gives some instances about sophisticated compliance verification cases.

- How to judge whether the curve of the historical command changing is legitimate. When the historical command data (those data with a numerical value) are recorded, then taking the time t as one axis and the command value as another axis, then a two dimensional curve can be drawn. This curve depicts how the historical commands changed, and can indicate how the new commands are likely to be. If a new command differs too much from the command trend from the command curve, then it is treated as illegitimate. There should be precise measure about allowable differences from the received command data and the expected value computed from the command curve. Common mathematical methods include computing the mean and variances of the command data. Since this kind of compliance verification is based on historical command data, artificial intelligence (AI) techniques can be used, which learns the historical command data, and judges whether a new command is legitimate. After verification, a legitimate command is then taken as a historical command.
- How to judge whether a command is legitimate in accordance with other data. For example, if a command indicates to increase heat when the environmental temperature (a kind of environmental data) is already higher than expected, then the command is likely to be illegitimate, and it is defined as legitimate when certain pre-defined rules and threshold values of some parameters are met.

When an authorized command is judge as illegitimate (when authentication is done successfully), the forthcoming action is taken depending on the requirement of the specific applications. In some cases, the command is still executed as legitimate, and a warning message is sent calling for human attention. In some other cases, the command may be hold waiting for human interaction, or the command is rejected with a warning message. In some special applications, an illegitimate command may be executed with a soft manner. For example, when a highly spinning device machine receives a command asking the spinning speed to increase 5000 revolutions

per minute (i.e., 5000 rpm), which seems to be illegitimate, and the device has to execute the command. In such case, a more soft manner to execute the command is to complete the command execution within a certain period of time. For example, instead of increasing the required rpm in time t it may increase 2500 rmp in time t, then increase another 1250 rmp in another time t, then increase another 1250 rmp in another time t. By this manner of execution, the command is executed in time $3t$. During the execution of the command, new commands may come to overwrite the previous command, even if the execution of the previous command has not yet been completed. This is particularly the case in car racing, where commands to control the car often change sharply.

In order to judge whether a command is legitimate, there should be a reference model. For example, assume that the reference value of a specific command is d_0, then there can be different judgement models such as follows.

1. Assume that the allowable changing range is Δd, i.e., when a command is within the range $[d_0 - \Delta d, d_0 + \Delta d]$, then the command is judged as legitimate. More specifically, given a command d, then d is judge as being legitimate if and only if $|d - d_0| \leq \Delta d$.
2. Assume that the reference value is a lower (or upper) bound, and the allowable difference is Δd. Then a command d is judge as being legitimate if and only if $0 \leq d - d_0 \leq \Delta d$.
3. There is no pre-defined range of allowable variation, then we can define the *deviation* as $\Delta d = \frac{d-d_0}{d_0}$, which can be a positive or negative value. It measures the difference between the command and the reference value, where the reference value may be an ideal value in theory.

In general, a more intelligent judgment would be based on how likely a command is legitimate (e.g. based on a percentage measure), instead of yes/no option. Such an intelligent judgment probably needs techniques in artificial intelligence. Artificial intelligence also helps to make correct judgment even if there is no pre-defined reference legitimate value.

11.9 On the Intrusion Tolerance Mechanisms in IoT Systems

In IoT applications, commands (as well as application data) may come from an authorized user, or come from a malicious attacker. Once an attacker intrudes into an IoT device (e.g., a personal mobile device or a master computer in an industrial control system) that can control some other IoT devices, then the command will be treated as being authentic, since the intruder can use all the security functionalities available from the intruded device, hence traditional authentication mechanism may not work properly.

Considering the fact that intrusion prevision and intrusion detection are practically difficult for IoT devices, particularly for those that with constrained resources, hence intrusion into some IoT devices is not a normal phenomenon. When treating

the cases where an authorized IoT device in an IoT system is intruded by an attacker, how to judge whether a command from that IoT device is legitimate needs techniques beyond traditional authentication. Intrusion tolerance seems to be suitable in this case. However, traditional intrusion tolerance techniques [19] do not suit IoT devices with constrained resources. Traditional intrusion tolerance techniques are built upon replication and secret sharing techniques, and the mechanism of intrusion tolerance is that the device being intruded is only part of a distributed system where a number of distributed devices work collaboratively to fulfill a task with security protection. Data replication is used in Byzantine fault model [8] and the state-machine based approach [13], targeting at protecting faulty replication of data. IoT devices however do not usually support collaborative work. Hence, the traditional intrusion tolerance techniques do not apply to the IoT systems.

It should be noted that the intrusion tolerance of IoT devices have the following assumptions, they are reasonable to OT attacks:

1. The intruder does not make any damage to the device being intruded. Intruder's activities such as collecting information of the intruded device cannot be prevented and is considered as having less harm to the IoT system.
2. The intruder tries to control some other IoT devices via the intruded device. This is particularly the case in industrial control systems when the intruded device is a master computer.

Upon these assumptions which are actually the specific features of IoT devices, we propose some different mechanisms to provide the behavior of intrusion tolerance. Our proposal is based on the assumption that there must be a manner of getting a special data (e.g., external data), or a special computing process (e.g., external computation), or a special communication channel (e.g., external computing), or a special means of control (e.g., human interference) that cannot be made available from the intruded device itself.

11.9.1 External Data Source

Needless to say, external data source means that the data is not available from the host machine, where the host machine is what is supposed to be intruded by an attacker. When the external data is a necessary part of control command, or is a necessary component to lunch a successful control command to the endpoint physical devices, then the intruder to the host machine is unable to forge a control command successfully, even if the intruder is assumed to have complete control over the host machine.

External data source is a common mechanism to provide IT security. For example, a password memorized by a user is a common way of external data. It is noted that, once the external data is used by the host machine, the data is stored in the machine's memory, which can be made available to the intruder if the intruder can access to the memory of the intruded machine. In order to strength the security

avoiding the possibility for the intruder to get the external data from the machine's memory, the external data should have the following properties.

1. **Computation infeasibility.** The external data is generated in a way that is independent of the host machine.
2. **Freshness.** The external data has the property of freshness, which means that, once the external data is used, it will become useless and cannot be used again;
3. **Independence.** There is no obvious connection between different instances of the external data, which means that, given some instances of the external data, it is infeasible to compute any other new data that can be used as valid external data.

11.9.2 External Computation

Like the external data, external computation is a functionality unavailable from the host machine, so that an intruder to the host machine cannot use the external computation, hence cannot forge a control command.

External computation is also a common security mechanism for IT security services. For example, in mobile communication systems, a SIM card is such an external device to a mobile equipment (ME). User authentication and voice/data encryption are all done by the computing inside the SIM, which is an external computing to the ME. The external computing often is composed with a kind of external data (e.g. encryption key).

11.9.3 External Communication

External communication is a different channel of data communication, this channel is independent of the host machine. Hence an intruder to the host machine cannot use the external communication channel. Different from the external data and external computation that interact with the host machine, the external communication is something independent of the host machine. External communication is an independent communication directly to the endpoint physical devices that are controlled by the host machine. The mechanism of external communication is that, the endpoint physical devices use the data from external communication to check the validity of the control command from the host machine. There must be some data generated externally as the content of the external communication that cannot be generated by the host machine, and the content of the external communication may or may not be based on the knowledge of the control command from the host machine.

In practical applications, there are systems that use multiple communication channels, only when multiple copies of the same data is received, the data is treated

as valid. This can be treated as a variant of external communication. For example, multiple communication channels are used in high speed rail systems. Although the purpose of using multiple communication channels in high speed rail systems is to ensure the reliability of data transmission, it also plays the role of security against intruder attack.

It is not necessary for the external communication to use a different physical communication line, both the external communication and the communication from the host machine can share a same communication channel. As long as the hose machine cannot control the external communication, intrusion tolerance can be achieved.

11.9.4 Human Interference

Sometimes data transmission may need human interference before the data arrives destination or before the data is accepted. The following two cases are such cases and have been practically used:

1. From sending part. When a control command is sent to an endpoint device, a confirmation made by human is made. This confirmation is to prevent the malfunction of the master control platform when sending a control command.
2. From the receiving part. When a control command is received by an endpoint device, a confirmation from human is made before the command is executed. This human confirmation is to double check the validity of the control command.

There can be different ways to realize human interaction. For example, by a physical switch or a soft switch. Software implementation of the confirmation risks the possibility of being manipulated by intruders, and has the advantage of using cutting-edge technology such as artificial intelligence (AI), which makes it an AI interference instead of human interference. It is obvious that human interference usually takes some time, hence is unsuitable to time-sensitive systems. However, time-sensitive systems can use AI interference instead of human interference.

Human interference is a mechanism to disable the illegal control even if an attacker (intruder) has complete control over the master control platform. However, this cannot guarantee the success of illegal control, because human have inherent weaknesses, for example, negligence, careless, limited knowledge and experience, these weaknesses may allow illegal control command to pass through the confirmation. Taken such factors into consideration, and given that human interference has a relatively long time delay, combined mechanism (e.g. external communication and AI interference) may provide a better way to tolerant intrusion.

11.10 An Architectural Construction of Intrusion Tolerance Scheme for IoT

Since intrusion to an IoT system is unavoidable, how to reduce the potential risks after intrusion is an important problem, particularly to the IIoT systems. If an IoT system has some capability of intrusion tolerance, then it can keep normal working even if an attacker has intruded into the system.

11.10.1 Traditional Intrusion Tolerance Techniques

The problem of intrusion tolerance for traditional information systems has been largely studied (e.g. [19]). Traditional techniques of intrusion tolerance include the following approaches:

1. Use plain data replication. To ensure a system to continuously providing correct services. This can be achieved by using the replication techniques, where all the data have a number of replicates, each copy is stored in a different place. As long as the intruder can only compromise and hence destroy a small number of replicas, the system can still work since it still has the data available.

   The disadvantage of plain data replication is that, it is often perceived to reduce the confidentiality of the data, because the leakage of any copy of data means that the data confidentiality is broken.
2. Split data into shares so that the data confidentiality can be properly protected.

   In order to overcome the disadvantage of plain data replication, an approach is to use the mechanism of secret sharing and threshold cryptography, where the data to be protected are split into different parts (called shares), each share is stored in a different place. When some of the shares are put together, the original data can be recovered. Secret sharing and threshold cryptography can provide services for this security requirement, and data confidentiality can be ensured as long as the intruder only gets a small number (less than the threshold value) of the shares.

The intrusion tolerance techniques described above are targeted at protecting the system from data loss and data stolen. However, the intrusion tolerance designed to thwart OT security attack is to avoid illegal control over the endpoint devices. A simple intrusion tolerance scheme for IoT systems is designed to demonstrate how the intrusion tolerance works.

11.10.2 A Practical Intrusion Tolerance Technique for OT Security Based on Architectural Design

This section proposes an intrusion-tolerance architecture targeting at OT security protection, where intruders are assumed not to damage the intruded master control computer, but try to illegally control other endpoint devices that are connected to it. The proposed intrusion-tolerance architecture is based on the existing industrial control systems with minimum modification.

Assume that there are n master control computer working together to control one endpoint device, or to control a number of endpoint devices. When a specific control command is sent to the endpoint devices, all the master control platforms send the same command within a short period of time. When an endpoint device receives the same control command from k or more master control computers, the command is executed. Otherwise, the endpoint device does not execute the command but to wait for more copies of the same command to come until timeout.

However, practically, the endpoint devices are often industrial controllers, such as PLCs or other more complicated machines. Many of the endpoint devices do not have the capability to add computational functionalities, even a simple matching operation (string matching in practical implementation) is a hard task to add onto the existing devices. Fortunately, the proposed intrusion-tolerance architecture can be realized by adding a combiner just before the endpoint devices. For example, when $n = 3, k = 2$, the proposed intrusion-tolerance architecture is composed of 3 master control platforms, and can be demonstrated in Fig. 11.1.

As is shown in Fig. 11.1, this instance of the proposed intrusion-tolerance architecture requires 3 master control platforms with the same or similar configuration and functionality of sending control commands. All the 3 master control platforms connect to the same combiner who then connect to the endpoint devices. When a control command is sent, all the 3 master control platforms send the same command, which arrives the combiner. The combiner checks if two identical copies of the control command is received, if so, then the control command is passed to the endpoint devices who are supposed to receive the control command. If one of the 3 master control platforms fails to send the correct control command, or it does not send anything, then the two copies of the control command sent by the other 2 master control platforms still enable the command to go through the combiner and arrive the endpoint device for execution. When any of the endpoint devices sends a feedback message, the combiner simply forwards the feedback message to all of the 3 master control platforms.

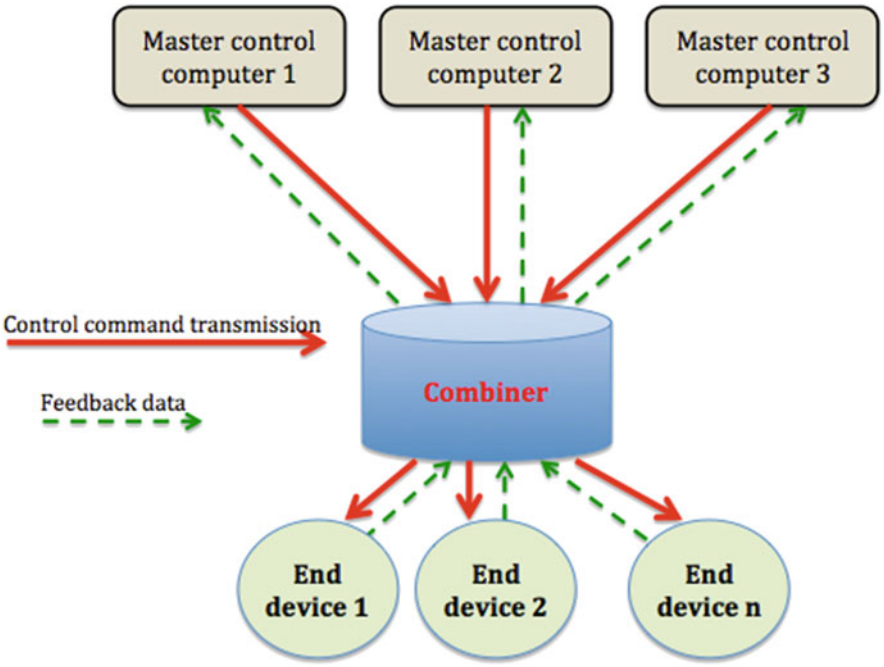

Fig. 11.1 A 2 out of 3 intrusion-tolerance architecture with a combiner

11.10.3 Security Protection Modes of the Intrusion-Tolerance Architecture

Considering the fact that intruders may have different capabilities to lunch attacks, there should be different ways of protection, here we call them security protection modes.

Mode 1 (plain combiner): As is shown in Fig. 11.1, when the endpoint devices transmit feedback message or application data to the master control computers, this message will be forwarded to all of the n upper level master control computers by the combiner. When a control command cmd needs to be sent to the endpoint devices, all the n master control computers are supposed to send the same command cmd to the endpoint devices via the combiner. If the combiner only receives less than k copies of the command cmd within the allowable time period (i.e., before timeout), then the command is discarded.

There are cases when the combiner receives less than k copies of the command cmd. One of the cases is that the combiner receives less than k messages, the other cases is that the combiner receives k or more messages, but the number of those message containing the same command cmd is less than k. There may be possibilities that the combiner receives k or more copies of another command

cmd' which is different from cmd, in the mean while the combiner also receives k or more copies of the command cmd, in this case, the command (cmd' or cmd) that k copies of them first come to the combiner will be sent to the endpoint devices for execution. If k copies of cmd' arrive earlier than k copies of cmd, then the endpoint devices actually execute a wrong command. In order to eliminate this case, k should be larger than $\frac{n}{2}$.

Mode 2 (security enhanced combiner): When the master control computers send control command cmd to the endpoint devices, security mechanism is added. For example, identity authentication of the sending computer, confidentiality and integrity protection of the command cmd, are common security services.

When such security services are provided, the combiner checks the authenticity of the message source, checks the integrity of the command message, decrypts the message, and stores the message. When k copies of a same message are received in such way, then the message is sent to the endpoint devices for execution. Compared with mode 1, mode 2 only adds security services for the command message and the source authentication of the message.

Mode 3 (randomized combiner): A random number R is agreed among the master control computers via some external channel before a control command is sent. The control command cmd together with the random number R are put into security protection, then sent to the endpoint devices via the combiner. Security protection may provide message source authentication, message confidentiality, message integrity, and message freshness services. When the message arrives the combiner, it checks the security protections and the correctness of R, then forwards the cmd to the endpoint devices when certain number of copies of R are received.

It is obvious that, the above security protection modes have different security requirements, with each mode having a higher level of security functionalities than the earlier ones.

11.10.4 A Preliminary Security Analysis on the Proposed Intrusion-Tolerance Architecture

As mentioned earlier, assume that there are n master control computers, and the combiner needs to receive k copies of the same command before the command is forwarded to the endpoint devices, then it requires that $k \geq \frac{n}{2}$, otherwise there is a possibility that the combiner receives two different command data each with k or more copies. On the other hand, k should be large enough to increase the security bound. We assume that $k = \lfloor \frac{n+1}{2} \rfloor$, where $\lfloor X \rfloor$ means the largest possible integer that is less than or equal to X.

Assume that an attacker has intruded into $t\,(t < k)$ master control computers. Then the attacker can produce t copies of any forged control command, hence the

combiner will not forward the forged control command to the endpoint devices. The other $n - t$ master control computers are assumed to work properly, and they can send $n - t$ copies of a control command to the combiner. Since

$$n - t \geq n - k + 1 = n + 1 - \lfloor \frac{n+1}{2} \rfloor \geq \lfloor \frac{n+1}{2} \rfloor = k,$$

the combiner will forward the command to the endpoint devices for execution.

If the number of intruded master control computers is $t \geq k$, then the attacker can send $t \geq k$ copies of any forged control command to the combiner, which will forward the forged control command to the endpoint devices for execution, which means the attack (in the sense of OT security) is successful.

Note that the above analysis is under the assumption of security protection mode 1. In mode 2, even if an attacker intruded into $t \geq k$ master control computers, the attacker has to run the required security functions for protecting the control commands, including the use of encryption and other keys. This process of security protection of control commands includes to find the proper keys for authentication, for message encryption (control command as a message), for message integrity protection (e.g. a MAC algorithm), and to run the authentication protocol, and cryptographic algorithms. Anything going wrong will result the attack not to be successful. This shows why mode 2 has higher security than mode 1.

For specific attacks where attackers are assumed to be professional that can do whatever a master control computer can do once they intruded into a master control computer, then if an attacker has intruded into $t \geq k$ master control computers, then the attacker can do what the t master control computers can, hence forged control commands can be executed by the endpoint devices, and the attack can be successful. However, with an additional random number generated externally and shared by the master control computers, the attacker will not be able to produce such a shared number among the intruded master control computers, unless there is a way to communicate among the intruded master control computers. If this is not the case (e.g., the intrusion is done by an internet warm), then mode 3 can provide security protection in the above case where mode 2 does not work.

Needless to say, mode 3 cannot provide absolute security against OT security attacks. However, in practical applications, mode 2 seems to be sufficient in many cases.

11.10.5 On the Reliability of the Proposed Intrusion-Tolerance Architecture

Apart from security, reliability is another problem to be serious considered in some IoT systems, particularly in IIoT systems. The implementation of the above proposed intrusion-tolerance architecture needs a combiner, which may reduce the

reliability of the whole system, since the reliability of the combiner itself as well as that of network connections may affect the reliability of the whole system.

Let the probability for a master control computer to be out of order for whatever reason be p, which is very small in practice, let the probability for the added combiner to break down be q, and let $q \leq p$ which is a reasonable assumption, since the combiner is much simpler than a master control computer. Then for the proposed intrusion-tolerance scheme, the system breaks down only when the combiner breaks down or $n-k+1 = n+1-\lfloor \frac{n+1}{2} \rfloor = \lceil \frac{n+1}{2} \rceil$ or more master control computers to be out of order. The probability for such a scenario to occur is

$$\rho = q + (1-q) \sum_{i=n-k+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}.$$

When $n = 3, k = 2$, the above equation becomes

$$\rho = q + (1-q)(p^3 + 3p^2(1-p)).$$

In order to compare the size of ρ with that of p, some specific values are given in Table 11.1, for simplicity, we let $q = p$, and hence $\rho = p + p^3(1-p) + 3p^2(1-p)^2$.

It is seen from Table 11.1 that the size of ρ is a little larger than that of p, which means that the reliability of the intrusion-tolerance architecture is not as good as traditional control model. Fortunately, it is possible to increase the reliability of the intrusion-tolerance architecture by increasing the reliability of the combiner. Table 11.2 gives some instances, where the size of ρ is required not larger than that of p by reducing the size of q.

It can be seen from Table 11.2 that, when the size of q is less than $0.97p$, the reliability of the intrusion-tolerance architecture is no less than that of the traditional control model. Practically, the reliability of the combiner can be made much higher than that of the master control computers, which reflect the size of q is significantly lower than $0.97p$, and can ensure the reliability of the intrusion-tolerance architecture.

Table 11.1 Comparison between the failure rate of the intrusion-tolerance architecture with traditional control model

p	ρ
0.001	0.001003
0.002	0.002012
0.003	0.003027
0.004	0.004048
0.005	0.005074
0.006	0.006107
0.007	0.007145
0.008	0.008189
0.009	0.009239
0.010	0.010295

Table 11.2 What size of q can ensure $\rho \leq p$

p	q	ρ	$r = q/p$(percentage)
0.0010	0.0010	0.0010	99.7%
0.0020	0.0020	0.0020	99.4%
0.0030	0.0030	0.0030	99.1%
0.0040	0.0040	0.0040	98.8%
0.0050	0.0049	0.0050	98.5%
0.0060	0.0059	0.0060	98.2%
0.0070	0.0069	0.0070	97.9%
0.0080	0.0078	0.0080	97.6%
0.0090	0.0088	0.0090	97.3%
0.0100	0.0097	0.0100	97.0%

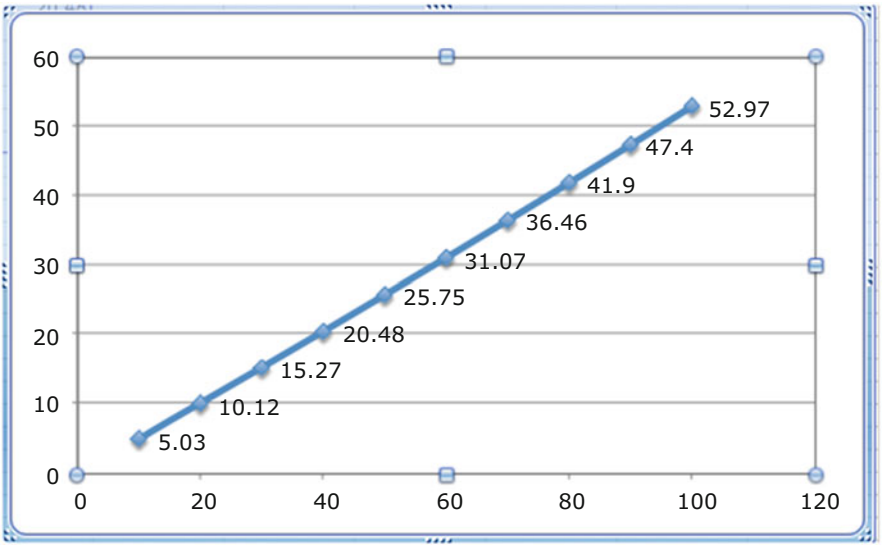

Fig. 11.2 How the value of ρ changes with the value change of p (The values of p and that of ρ are enlarged by 10,000 times)

For convenient description, set $q = 0.5p$ as an example. Then we have

$$\rho = \frac{p}{2} + p^3(1-p)(1-\frac{p}{2}) + 3p^2(1-p)(1-\frac{p}{2}).$$

With this representation, how the value of ρ changes with the value change of p can be shown in Fig. 11.2.

Figure 11.2 shows that, when we set $q = 0.5p$, the value of ρ is roughly half that of p, which means that, the chances for the intrusion-tolerance architecture to go wrong are about half of those for the traditional control model to go wrong.

11.11 Some Suggestions on Security Protection of IoT Systems

With respect to many features of OT security which are typically different from those for IT security, techniques for security protection are also different. The actual effect of security protection for OT security not only depends on the techniques, it also closely depends on relevant policies. Many of the techniques for intrusion tolerance are closely policy related, for example, the external data, computation, and communication, are all policy related techniques.

If a system is physically isolated from the public networks, it cannot completely avoid network attacks, including IT attacks and OT attacks. One possible way to transfer internet warms into such an isolated system is via USB disks. Such a way of infection can be understood as occasional network connection. In this sense, absolute isolated systems from the public networks do not practically exist.

We recommend that security protection mechanisms should be considered from the aspects of network exposure, identity authentication, intrusion tolerance, system redundancy, and system recovery/replacement. It will be seen that, many of the recommended security protection mechanisms are trivial and practical.

1. **Reduce unnecessary network exposure.** Many IoT systems do not have to be connected to the network all the time. For example, electricity meter reading, as a specific IoT application, requires a few times meter reading per day, and each time of meter reading requires a few seconds. Most of the time, the meters as IoT endpoint devices, do not need to connect to the network at all. Given the fact that the first step for many network attackers (e.g. internet warms) is to find attacking targets by scanning the Internet. If the IoT device breaks down its network connection when the network port is scanned by an attacker, the attacker is likely to skip attacking this system, because the scanning process does not reveal the existence of this device.

   Practically, apart from scheduled time for meter reading, there may be cases when a meter reading is temporarily needed, so the meters need to be connected to the network from time to time, and then disconnect the network after a short period of network connection. Assume that when a meter connects to the network, the connection will remain for t seconds, then it disconnects the network and waits for T seconds till the next time of network connection, where T is practically much larger than t. If a command is sent to the meter for temporary meter reading, the command sending has to be kept for $T + t$ seconds to ensure it reaches the meter. For example, if $t = 5$ s, $T = 55$ s, then the chances for the meter to be connected to the network is $t/(T + t) = 0.083$, which means that, the chances for a network scan to find the meter is only 8.3%, instead of 100% if the meter is connected to the network all the time.

   For a more practical policy of network disconnection, individual cases have to be analyzed. The general principle is that there should be a reasonable balance between network exposure (the less the better) and the network connection (meet the application needs).

2. **Enable intrusion tolerance.** With whatever security protection, it should be assumed that intrusion to an IoT system is possible. This is particular the case when the security protection for IoT systems is of limited power given the resource constraint feature of some IoT devices. Once an IoT system is intruded by an attacker, the best way is to minimize the damage by the attacker. Targeting at OT security protection, the most effective method is intrusion tolerance mechanism.

   There have been many techniques for intrusion tolerance. However, for the protection of OT security in an IoT system, the intrusion-tolerance mechanism proposed above is recommended, because the proposed intrusion-tolerance mechanism is designed for the specific situation of OT security in IoT systems.
3. **Maintain sufficient system redundancy.** In terms of system redundancy, it is for improving the reliability of the system operation. System redundancy includes the following respects:

   Device redundancy: during the sensor data collection, the malfunction of small number of the sensors should not affect the overall usefulness of the collected data. However, the redundancy is not simply to produce replication of the collected data, but different data samples for the environment;

   Network redundancy: during the data transmission, small number of broken connection should not affect the connectivity of the network which can fulfill the task of data transmission;

   Data storage redundancy: important data is stored with certain level of replication or similar mechanism, so that when some of the stored data is unavailable for whatever reason, the original data can still be recovered. The techniques for the above redundancies need further research.

11.12 Summary

The security problems in IoT systems are considered to include IT security and OT security. Although the problem of OT security has been noticed in industry, it lacks formal definition and systematic study. This chapter gives a preliminary study on the OT security problems and protection techniques, pointing out the differences between OT security and IT security. It is seen that the OT security is mostly from IIoT systems. As a technique for OT security protection, an intrusion-tolerance architecture is proposed, which can be used in IoT devices with constrained resources.

References

1. A. Bogdanov et al., PRESENT: an ultra-lightweight block cipher, in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ed. by P. Paillier, I. Verbauwhede. Lecture Notes in Computer Science, vol. 4727 (Springer, Berlin, 2007), pp. 450–466
2. H. Boyes, B. Hallaq, J. Cunningham, T. Watson, The industrial internet of things (IIoT): an analysis framework. Comput. Ind. **101**, 1–12 (2018). https://doi.org/10.1016/j.compind.2018.04.015
3. R. Chaudhary et al., SDN-enabled multi-attribute-based secure communication for smart grid in IIoT environment. IEEE Trans. Industr. Inform. **14**(6), 2629–2640 (2018)
4. X. Du, Y. Xiao, F. Dai, Increasing network lifetime by balancing node energy consumption in heterogeneous sensor networks. Wirel. Commun. Mob. Comput. **8**(1), 125–136 (2008)
5. W. Eisenbarth, S. Kumar, A survey of lightweight-cryptography implementations. IEEE Des. Test Comput. **24**, 522–533 (2007)
6. D. Hough, IET: cyber security in modern power systems: IT and operational technology integration, in *IET Cyber Security in Modern Power Systems*, June 2016. https://doi.org/10.1049/ic.2016.0047
7. D. Kushner, The real story of stuxnet. IEEE Spectrum **50**(3), 48–53 (2013). https://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet. Accessed 10 July 2020
8. L. Lamport, R. Shostak, M. Pease, The Byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
9. P.I.L. Lehlogonolo, Determining the performance costs in establishing cryptography services as part of a secure endpoint device for the Industrial Internet of Things. MSc Dissertation, University of Pretoria, 2018
10. V. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, N. Shroff, A minimum cost heterogeneous sensor network with a lifetime constraint. IEEE Trans. Mob. Comput. **3**(3), 1–12 (2004)
11. S. Pinto, T. Gomes, J. Pereira et al., IIoTeed: an enhanced, trusted execution environment for industrial IoT edge devices. IEEE Internet Comput. **21**(1), 40–47 (2017)
12. M. Saleem, G.A.D. Caro, M. Farooq, Swarm intelligence based routing protocol for wireless sensor networks: survey and future directions. Inf. Sci. **181**(20), 4597–4624 (2011)
13. F.B. Schneider, Implementing fault-tolerant services using the state machine approach: a tutorial. ACM Comput. Surv. **22**, 299–319 (1990)
14. J. Shen et al., Lightweight authentication and matrix-based key agreement scheme for healthcare in fog computing. Peer-to-Peer Netw. Appl. **12**(4), 924–933 (2019)
15. B. Vigliarolo, WannaCry: a cheat sheet for professionals. https://www.techrepublic.com/article/wannacry-the-smart-persons-guide/. Accessed 10 July 2020
16. Z.J. Wang, K.W. Li, W.Z. Wang, An approach to multi-attribute decision making with interval-valued intuitionistic fuzzy assessments and incomplete weights. Inf. Sci. **179**(17), 3026–3040 (2009)
17. W. Wu, L. Zhang, LBlock: a lightweight block cipher, in *Proceedings of the 9th International Conference on Applied Cryptography and Network Security*. Lecture Notes in Computer Science, vol. 6715 (Springer, Berlin, 2011), pp. 327–334
18. W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, I. Verbauwhede, RECTANGLE: a bit-slice ultra-lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. **58**(12), 1–15 (2015)
19. W. Zhao, Intrusion tolerance techniques, in *Encyclopedia of Information Science and Technology*, 4th edn. (IGI Global, Hershey, 2018), pp. 4927–4936

Chapter 12
A Comprehensive Set of Security Measures for IoT

By scrutinizing minor differences between different security requirements, this chapter presents a set of security measures for IoT. The proposed set of security measures categorizes security requirements in different aspects, including physical security, execution environment security (e.g., operation security and software security), network security, data security, key management security, and availability related security. Security test methods are discussed. There are general security test methods, and specific security test methods for specific security measures. It is pointed out in this chapter that, a same security requirement can be achieved using different methods, and a same method can be used to gain different security services. A small example is presented to show that, a simple encryption can achieve data confidentiality service, as well as data source authentication, data integrity, and data freshness services at the same time. This example can be a reminder of that, security test should focus on the security services instead of functionalities.

12.1 Introduction

The Internet of Things (IoT) has been widely used in many different applications, including industry, commercial, healthcare, and residential. IoT can link the real world with the information systems, forming a cyberspace by techniques from cyber-physics systems and some other cutting-edge techniques. IoT security, however, has many technical challenges not having been properly solved, some of the problems in IoT have not yet been properly addressed. This chapter describes some characteristics of the security requirements in IoT systems, presents a comprehensive list of possible security measures for different kinds of IoT systems, mainly focusing on the perception layer of IoT systems, i.e., the logical layer mainly for raw data collection and command execution in IoT systems. The listed security measures are not meant to be necessary for all IoT systems, and by no means complete.

C.-K. Wu, *Internet of Things Security*, Advances in Computer Science and Technology, https://doi.org/10.1007/978-981-16-1372-2_12

It is noted that, some of the security measures conflict with each other, so proper compromise should be made before a security solution can be made practically. Comments are given with respect to how to test the security measures. It is expected that the security measures in this chapter can serve as a resource of reference for security designers, security practitioners, and security standard makers for IoT related applications. For a specific IoT system, a small set of the security measures should be sufficient. Many security measures need to be made more precise and concrete for specific IoT applications.

12.2 The Special Features of IoT Security Versus Traditional IT System Security

Recently, many traditional security protection techniques are adopted in the IoT applications. Industrial standards, national standards, and international standards for the IoT security are gradually made available. However, most of the standards did not consider the characteristics of IoT systems properly, and hence try to implant the security requirements in traditional information systems into IoT applications. Moreover, many security requirements specifically for IoT systems are not addressed properly in those standards. The main reason is that the differences between IoT systems and traditional information systems have not been clarified properly.

Our preliminary study on IoT security found that the security of IoT systems has the following characteristics:

1. It covers information security services in the respects of confidentiality, integrity, and availability.
2. It is often applied in the resource-constrained environments [5], particularly for an IoT device with low processing capability and battery power. In this case, the security solutions should have the feature of being *lightweight*.
3. Apart from traditional information security services, it also covers operational security (i.e., OT security). For example, stability, reliability, and intrusion tolerance, are all regarded as OT security.
4. IoT security considers the protection of the IoT system from external attacks, and the protection of IoT devices from being involved in attacking other network resources.

The OT security deals with operation, which is closely related to the safety, while IT security deals with information, which is about security. Safety and security have close relationship in industrial control systems [6]. This chapter mainly considers the information security, even the OT security is trigged by the information processing.

IoT security measures are practical guidance to the IoT application system design and operation. Grabovica et al. presented a survey of the security measures for IoT [4], and the security measures do not have a good coverage.

Considering the security requirements from industry application aspects, this chapter presents a list of security measures, and with brief comments about how to test those security measures. Some of the security measures may conflict to each other, in such cases, an appropriate compromise between the security measures should be made in the establishment of such security requirements. Many of the security measures seem are beyond the necessity for most of the IoT systems or IoT devices, this situation is normal, because the purpose of this list is meant to provide a source of reference, which is supposed to have a fairly good coverage, not meant to have all of them implemented.

It is noted that one special feature of IoT systems is the OT security, which has substantial differences from IT security.

12.3 Security Measures and Test Methods for IoT Perception Layer

Based on the security architecture as shown in Fig. 2.1 it is known that, the IoT perception layer is the main component in IoT architecture, because it is the critical part that links the information network to the physical world, and has some essential differences from traditional information systems. Many constraints of IoT devices in the perception layer makes the security measures different from that being expected in traditional information systems.

In order to have a good coverage of security measures, this chapter considers the following security measures in the perception layer of IoT: physical security measures, security measures in execution environment, security measures in network, security measures in data security, security measures in key management, and security measures in availability. Those security measures do not necessarily cover all the security requirements for IoT applications in all industry and consumption, and are not necessarily feasible in many IoT applications. Irrespective of our effort, there may be some special security requirements needed in a specific IoT application but not covered in this chapter, and we try to eliminate such cases as much as possible. It is not surprising to find that a number of security measures are not necessary for many IoT applications.

Nevertheless, this chapter tends to provide a comprehensive list of security measures for IoT, particularly the perception layer of IoT, and presents possible test methods for the security measures. Some of the test methods are simply general guide with little or no technical components, and some of the test methods do not really need technical components.

12.3.1 Physical Security Measures

The physical security are mainly about the protection of the IoT devices when they are working in a specific environment. Physical security measures include protection of IoT devices from illegal physical action made by human, protection of IoT devices from being affected by a wrong environment, and the protection of the environment from unacceptable effect caused by the IoT devices.

12.3.1.1 Security Measures for IoT Devices

***Measure*-1** The IoT devices should have protection from being stolen.

Many IoT devices are installed in open areas with easy access from public, particularly the low-cost IoT devices. If such IoT devices process confidential data, even if they have encryption algorithms, the biggest security threat is that they are taken away and examined in a laboratory, and their secret information such as an encryption key can be revealed. The successful attack on one of the IoT devices may have direct security threat on many other IoT devices, and may further have the security threat for the equipment communicating with those IoT devices. Hence for IoT devices with import applications, regardless of the cost of the devices themselves, if the data that will be processed by those devices are of high value, it is important to have some physical protection against being captured.

Rationale: IoT devices can be stolen for a number of reasons, e.g. some people are just curious, or have bad mood, or want to do experiments or take examination on them. Low-cost IoT devices do not care about the low possibility and low risk of being stolen, however, important IoT devices should take some action to protect them from being stolen.

There are many ways to protect IoT devices from being stolen, Some of the protection methods are as follows:

(i) Physically fix the IoT devices to a firm base, for example, put an IoT device into a metal box and then lock the box.
(ii) Put the IoT devices under surveillance.
(iii) Put the IoT devices in a special area with security protection, where unauthorized people can hardly get in.
(iv) Put the IoT devices in the places which are difficult to reach or find. For example, put the IoT devices on the top of a power pole, or put them in a wild field.

Test method: The most practical method of test on the above security measure is the human on-site examination test as described in Sect. 12.4.4.

***Measure*-2** IoT devices should have protection against cloning.

Attackers have little motivation to clone sensor devices in IoT systems. However, they are interested in cloning RFID tags. For the RFID tags without security protection, it is easy to clone them, which can simply be done by reading a tag's

ID and then write it into a blank RFID tag. If the RFID tags have some security protection, attackers may still be able to clone them, depending on the security level of the protection, the knowledge that the attackers have, and the effort that the attackers make in the attack. When such RFID tags are used as access cards in a keyless access system (e.g. RKE), cloning such tags may cause a big security problems.

Rationale: RFID tags cloning is a common phenomenon. Many RKE systems face cloned access tags.

There are many protection methods against RFID tags cloning, some of the methods may cause large cost, while others have relatively less cost. Security protection methods for RFID tags against being cloned include the following:

(i) To use individualized communication protocol between the RFID tags and the reader.
(ii) To encrypt the data transmitted between RFID tags and the reader.

The first method has been used, and the security protection is limited, because knowing the specific communication protocol would be able to clone the RFID tags. The second method has higher level of security than the first method, however the implementation cost is high, and has not been a popular method yet. From a long run, the second method has obvious advantage than the first method in protection of large scale RFID from cloning.

Test method: To test whether an RFID tag has protection against being cloned, it needs first to check whether the protection is appropriate in principle, then attempt to make a clone by using an equipment and see whether it works. The cloning test is a kind of attack simulation test as described in Sect. 12.4.6, it should include traditional methods as well as new methods, which varies from time to time.

***Measure*-3** IoT devices should have certain protection against DoS attack.

The DoS (denial of service) attack is a common attack in traditional information systems [8]. The purpose of DoS attack is to make the target system lose the capability of providing normal services. A traditional DoS attack is to exhaust the resources (e.g., CPU, memory, or network bandwidth) of the target server. This strategy also applies to the DoS attack on IoT devices.

Apart from the traditional approach of DoS attack, another mechanism of DoS attack on IoT devices is to consume the energy of the target device. It can be done by consistently sending queries, connection requests, or application data to the target device. The process of receiving and handling the message consumes energy, which may eventually run off the battery of the target device, even if the messages are verified as illegal. This kind of DoS attack is specific to IoT devices, it does not necessarily need to send a large amount of messages in a short period of time, instead, consistently sending messages to the target device, which may last for a relatively long time, say a few days, is the feature of such a DoS attack on IoT devices. The attack may be split into different intervals, which do not reduce the effect of the energy consumption purpose. When the target IoT device runs off the battery, the attack is successful.

Although to resume the working status of an IoT device when it runs off the battery seems to be easy, it is a problem when large amount of IoT devices run off the battery. This manner of DoS attack is meant to target at a large amount of IoT devices. Sometimes even changing a battery of an IoT device is not easy, because many IoT devices are designed to have the same lifetime with an integrated battery, which means that, when the battery runs off power, the device also reaches its lifetime.

Rationale: Traditional methods for information systems to thwart DoS attacks is to check the processes suspecting to be DoS attacks, and try to end those processes. This action may also thwart some normal connection attempts. However, normal connections are likely to resume after some time, so the service can be resumed soon after the DoS attack stops.

With respect to IoT devices, they may not have the capability to check the execution processes. An effective way of reducing the effect of DoS attack is to use the mechanism of sleeping/dormancy. More specifically, when no authorized messages come to an IoT device for a certain period of time, the device goes to sleep. By doing this, the IoT devices are immune against DoS attack when they are in the state of sleep/dormancy.

Test method: To test whether an IoT device has certain protection against DoS attack, an effective way is the attack simulation test as described in Sect. 12.4.6, The attack simulation may vary from time to time, requiring the protection methods to be robust against different DoS attacks.

***Measure*-4** Important IoT devices should have backups available.

IoT devices are not just the small ones, they include normal PCs or even work stations. For the consistence of description, we still call them IoT devices. For the IoT applications in industry, namely industrial Internet of Things (IIoT), a master control station controlling many PLCs or other control devices is treated as an IoT device, which is obviously an import such device. Backups for the important IoT devices are necessary in many applications. It is noted that whether an IoT device is treated as being important is subject to its role in a specific application. The backup of an IoT device is meant to replace the IoT device when the device does not work properly for all reasons. The backups of IoT devices are not simply the replacement of a physical machine, they include appropriate configuration with software updates as well as necessary updated data.

Rationale: This security measure is mainly targeted at Industrial Internet of Things (IIoT), where many master control stations practically have backups.

Test method: To test whether an IoT device (treated as being important) has a backup, it is to check the existence of such a backup machine, and to see if it can replace the IoT device to work when the IoT device is turned off (or set aside). This test can be done by human on-site examination test, as described in Sect. 12.4.4.

***Measure*-5** Key IoT devices should have a hot-standby backup.

A hot-standby of an IoT device (in the case of a computer or workstation) is a redundant machine in a system, where one IoT device runs simultaneously with

an identical primary IoT device. Upon failure of the primary IoT device, the hot-standby device immediately takes over the functionalities of the previous primary IoT device.

A hot-standby is required in some industry control systems, which can be extended to IIoT. Taking IIoT as a special kind of IoT, hence the requirement for a hot-standby makes sense, and this security measure is needed only for important devices such as a master control station or the surveillance system of a prisons.

Rationale: This is a higher level security measure compared with ***Measure*-4**. Practically many IIoT systems do have hot-standby system. When this security measure is met, it means that ***Measure*-4** is also met.

Test method: The test on this security measure can be done by human on-site examination test as described in Sect. 12.4.4. In the test, the system of the main device is switched to the standby device, and check whether the system works as usual.

***Measure*-6** The identity of any IoT device should be unique.

Even though the identities of IoT devices in one application environment are unique, if a same identity appears in different applications, and the wireless signals interfere with each other, identity collision is also possible. In order to avoid replicate identities of IoT devices appear in different applications with possible signal interference, some mechanism should be applied.

The safest way to avoid identity collision is to define identities to ensure world-wide uniqueness. If this is inconvenient, then an alternative approach is first to define network identities to avoid collision, and then to ensure the uniqueness of identities in the internal network of each application, and the identities of IoT devices are combined with the network identities within the specific application. By doing that, there will be no collisions to the compound identities.

It is noted that in practical applications, the need for world-wide uniqueness of identities may be inconvenient, and is not necessary. If network identity collision becomes very unlikely, then the method of compound identity definition is acceptable.

Rationale: The uniqueness of the identities of IoT devices is to avoid possible collision. If the identities of IoT devices are defined in a way which ensures the world-wide uniqueness, then the identity uniqueness can be ensured. Otherwise, even though the IoT devices in one application system each has an unique identity, there still have chances that identity collision occurs with other application systems.

Test method: The test on this security measure can be done by checking the following: (1) to check whether the IoT devices have globally unique identities or (2) to check the uniqueness of the identities within the application system, and to check the probability that different application systems have identity collision. This is a coarse-grained test, and more measures about the acceptable level of probability should be made specific before the test of this security measure can be practically possible.

12.3.1.2 Security Measures on Effects from Environment to Equipment

***Measure*-7** The environment for IoT devices to work should not cause damages to the IoT devices.

A large amount of IoT devices are sensors of different types, such as sensors for temperature, moisture, light, pollution, etc. Those sensors should be installed in their working environment, and sometimes should be fixed somewhere so as to capture the correct environmental data. The installation should take careful consideration to reduce the risks of causing damages to the devices. For example, a sensor measuring the speed of a liquid should be fixed properly, while not allowing the sensor to be flown away, and not to cause damage to the sensor.

Rationale: With respect to the physical security of the IoT devices, it cannot be as that of traditional information systems which have high requirement on the environment to ensure their normal working condition. However, inappropriate installation of IoT devices may affect their proper working condition. This security measure is one of the typical security requirements specifically to IoT devices.

Test method: The test on this security measure can be done by human on-site examination test as described in Sect. 12.4.4.

***Measure*-8** The environment for IoT devices to work should ensure that the IoT devices would not shift beyond allowable range.

Similar to the cases of ***Measure*-7**, IoT devices should be installed properly, so that the environment would not cause them to shift beyond allowable range, ensuring that correct environmental data can be collected. In appropriate way of fixing the device may cause its damage.

Rationale: The environment where IoT devices work may cause all kinds of problems. This security measure is to ensure that the IoT devices are in the place where they can capture the correct environmental data.

Test method: The test on this security measure can be done by human on-site examination test as described in Sect. 12.4.4.

***Measure*-9** The environment for IoT devices should not affect the normal working condition of the IoT devices.

If an IoT device is installed in a wrong place, then it may not be able to capture the correct environmental data as expected. For example, if a temperature sensor is exposed to the sun, then it may not get the correct air temperature. Circumstances as such should be avoided.

Rationale: The quality of data is not only concerned with the reliability of the IoT devices, but the appropriate places where they work properly.

Test method: The test on this security measure also can be done by human on-site examination test as described in Sect. 12.4.4.

***Measure*-10** The environment in a changing condition should not affect the proper working condition of the IoT devices.

Simply to ensure the normal working condition of the IoT devices is not enough. Sometimes the working condition of IoT devices may change from time to time due to the changing condition of the environment. For example, a sensor for air

temperature may work properly in the morning, while it becomes being exposed to the sun in the afternoon and hence does not work properly. Such kind of changing factors may cause the environmental data corrected by the IoT devices to be incorrect.

Rationale: This is an extended security measure of ***Measure*-9**, where the environmental condition change may affect the appropriate working condition of the IoT devices. The condition change is not only caused by time, it may also be changed by changing seasons.

Test method: This security measure is to ensure that the normal working condition remains when time changes. The test on this security measure can be done by human on-site examination, and should last for a reasonable period of time, or the examiners are able to reliably foresee the possible environmental change when time changes, and to ensure the proper working condition along the period of time for consideration.

Sometimes the test only needs to be conducted at certain specific points of time instead of the whole period of test time. Nevertheless, the human on-site examination test would be appropriate for such test.

12.3.1.3 Security Measures on Effects from Equipment to Environment

Here the effect on the environment is mainly from the point of view of people's daily life, with little about the environment itself. Although such security measures are not related to information technology (IT) or operational technology (OT), it is important to take them into consideration when practically design a secure IoT system.

***Measure*-11** The effect of IoT devices on the environment should be kept to an acceptable level.

IoT devices are not just small devices. In industrial IoT, large industry equipments can be classified as IoT devices in general, and some equipments may generate big noise, emit poison material (e.g., liquid, gas, dust), which cause bad effect on the environment, particularly when the environment is near residential areas.

Some IoT devices may risk being blown off from a high place by strong wind. This kind of factors come from the installation of such devices. Regardless whether the effect of IoT devices on the environment comes from the IoT devices themselves, or from their installation, such effect has to be kept within acceptable level.

Rationale: The working condition of IoT devices may have effect on the environment, and the installation of IoT devices may also have effect on the environment. It is necessary to measure such effects.

Test method: The test on such security measure is usually done be human on-site examination as described in Sect. 12.4.4. This is a coarse-grained test, and other standard measures (such as the standard specification about acceptable level)

should be made available before the test on this security measure can be done practically.

A particular case is when the IoT devices are installed in a wide area, or to work in a wild field, this security measure is usually treated as being met in this case.

***Measure*-12** There should be a way to switch off an IoT device.

There are different ways to switch off an IoT device, and doing it physically is usually more reliable. The following are some common methods to switch off an IoT device:

1. Physical switch. An IoT device can be switched off by pressing such a physical switch. This is the most reliable method to switch off a device. However, this way of turning off the device is inconvenient if the device is in a place not easy to reach, for example, a sensor fixed on top of a power pole, or the device is too small to have such a physical switch.
2. Remove the battery, if possible. Some IoT devices are equipped with a battery. However, many such batteries are integrated with the IoT devices and can hardly be removed. Even if it is possible to remove the battery, it may be inconvenient to do so, when it is inconvenient to physically reach the IoT devices.
3. Switch off via network operation. Another approach which suits the operation on a bunch of IoT devices is to switch off the designated IoT devices via a special network operation. For example, a special network port is designed for special commands only, and a special format of command from the special network port is designed to force the devices to switch off. It is not necessarily to shutdown the devices, putting them into sleep can solve may problems. By doing that, these devices can be waken up at a later time when so needed.
   This method is convenient to operate, but may not work properly if the device is under DoS attack, or the intruder rewrites the relevant internal logic of the IoT device.

Rationale: IoT devices may not work properly for all reasons. For example, they may be illegally controlled by intruders, or may be suspected to risk a bad intrusion. In such cases, the IoT devices should shutdown.

Physically switch off the device is the most reliable and secure method, and it costs more as well (the cost of the physical switch, and the cost of human operation). Networked switch is more convenient, and it is less reliable.

Test method: The test on this security measure is a simple task. The test can be done by simply checking the existence of a physical switch or removable battery, or the operation to put the devices to sleep. This is an existence examination test as described in Sect. 12.4.1, which should take human on-site examination test as described in Sect. 12.4.4.

***Measure*-13** IoT devices should be able to re-install/recover its initial state by a hard or soft switch, or by an online command.

A hard switch is like to press the “on” button for certain period of time (e.g. a few seconds), and a soft switch suits the case when there is a touchscreen on the device.

The device initialization by an online command is more convenient, which suits the IoT devices installed in places inconvenient to reach, and can operate the initialization for a large number of IoT devices.

Some initial setup values have to be stored in some non erasable storage of the device, so that the device initialization can be done.

Rationale: The initial state of an IoT device is usually set by its manufacturer. In many cases, it is necessary to re-install the initial state of an IoT device. For example, when an IoT device is suspected to be controlled by intruders, or the device is reset for different use.

Test method: The test on this security measure is to examine the existence of the physical or soft switch. This is an existence examination test as described in Sect. 12.4.1, which should take human on-site examination test as described in Sect. 12.4.4. To test whether there is an online command that can initialize the IoT device, the test is simply to proceed such an initialization process.

Note that the IoT device to be tested should be with a state different from its initial state, and there should be a way to verify whether the IoT device is indeed in the initial state.

***Measure*-14** The process of online initialization of IoT devices should be protected by a key (e.g. a password) to ensure the authentication of the initialization command.

For small IoT devices, to recover its initial state by a hard switch or a soft switch is not practical, because they do not have plenty room for a hard witch, no room for a screen to operate a soft switch either. In such a case, if reuse of old devices is sometimes better than to use brand new ones, then a method to recover the initial state of the IoT devices is helpful. The most practical method is to do this online, while there should be an authentication mechanism to confirm that the command for initial state recovery is authentic.

Rationale: If the online initialization process does not have security protection, then attackers may forge such initialization commands to harass the normal use of the IoT devices.

Test method: The test on this security measure is to do the initialization process and see how the security protection works. If the initialization process is protected by a password, then attempt to use wrong passwords will result in initialization failure. This test is a correctness examination test as described in Sect. 12.4.2.

If the process is protected by a key which was previously defined, then the command data needs to be checked to ensure that the data complies with the security mechanism. This is an existence examination test as described in Sect. 12.4.1.

***Measure*-15** IoT devices should not contribute to DDoS attacks.

In October 2016, there was a big DDoS attack in American, where large amount of IoT devices were involved. Those IoT devices as well as many other network equipment contributed to the DDoS attack, because they became nodes of a botnet controlled by intruders.

Although trying to reduce the chances of malicious intrusion is what IoT devices ought to do, it is not possible to completely avoid intrusion, particularly for those devices with constrained resources. However, it is possible to eliminate the IoT devices not to be part of a botnet. This can be done by fixing the outgoing IP addresses with which an IoT device is supposed to communicate. If such a setting cannot be re-written by intruders, then the security measure can be met.
In order to do this, a possible approach is to set a special communication port for setting up the outgoing IP addresses. For example, if the port is for short distance wireless communications such as WiFi, then it would be difficult for network intruders to manipulate with.
Rationale: Conventionally, security consideration mainly cares about the protection of the target devices against being attacked by others, it hardly considers the possibility of attacking others. For IoT devices, this security measure is important, because the amount of the IoT devices in the future is a huge number, and making some of the IoT devices to form a powerful botnet is perhaps the most attractive usage by attackers.
Test method: The test on this security measure can be done by checking whether the outgoing IP addresses are defined, and this configuration cannot be changed by network connection to the device. This is a kind of validity test as described in Sect. 12.4.5.

12.3.1.4 Advanced Security Measures

Advanced security measures are only designed for security chips, or MCUs executing cryptographic algorithms. When such chips are used in IoT devices, since those IoT devices are easy to capture by attackers, attack on such devices can be conducted in a laboratory, and hence is under a high level of security threat. These security measures are required for applications with high security requirements.

***Measure*-16** IoT devices should have some mechanism to resist against side-channel attack.
Side-channel attack is a special attack usually with physical interaction with the device under attack. Passive side-channel attack is to listen the signal emission of the device in the process of executing a cryptographic algorithm, from which the power consumption curve can be drawn, which can be used to reveal some secret information of the cryptographic algorithm. Active side-channel attacks involve signal injection into the device to cause errors during its cryptographic algorithm execution, and are more powerful, and are harder to operate as well.
Rationale: If an IoT device processes important data, or is connected with controlling an important machine, then a cryptographic algorithm is likely to be used. The key to the cryptographic algorithm is not easy to get by attackers, but attackers can get such a physical device and conduct side-channel attack trying to recover the key.

Test method: It is not easy to test whether an IoT device has resistance against side-channel attack. A practical test method is to check whether relevant mechanism has been used. For example, chip packaging technique may provide some resistance against side-channel attack, software flow balancing technique, some software obfuscation technique, also have resistance against side-channel attack. If any of such mechanisms are applied, it is treated as satisfying this security measure, subject to the level of this security requirement.

This test is a suitability examination test as described in Sect. 12.4.3. In certain applications, this security measure needs more specific description, and the corresponding test method needs to be more specific as well.

***Measure*-17** IoT devices should have certain level of resistance against physical dissection.

Physical dissection of security devices, particularly security chips, may result in some security data in the memory being revealed, such as seed keys and initial values, states of counters, etc. These data may enable attackers to lunch successful attacks.

It should be noticed that physical dissection attack involves a high cost, and the probability of successful physical dissection depends on how the chip is protected and the facilities that the attackers have. Unless the target device is related to critical applications, the chances for such an attack are rare.

Rationale: Side-channel attack may or may not involve the dissection of the target IoT device. Dissection of the IoT device can help the side-channel attack as well as other attacks, and the dissection process incurs considerable cost, hence the security measure is required only in some rare cases.

Test method: In order to protect such attacks, many security chips have applied security protections against physical dissection. The test on this security measure can be done by checking the appropriation of the physical packaging of the chips, which is a suitability examination test as described in Sect. 12.4.3.

12.3.2 Execution Environment Security Measures

Physical environment is the outside of IoT devices, while the execution environment is an inside factor directly related to information security. The execution environment mainly means the operating system and the application software. There are a range of different operating systems for IoT devices, from chip operating system (COS) for chips, to traditional Windows/Linux operating system, or even Solaris, and other operating systems for IoT devices (such as Android). With respect to the COS or some embedded operating systems, their security protection may not be on the same level as that for other operating systems for traditional computers. For this sake, the security requirements on the execution environment of IoT devices mainly consider the devices with constrained resources.

There have been many mature techniques in operating system security. These techniques are designed for traditional operating systems, which include the

functionalities such as account management, access control, security auditing. The security functionalities for devices with constrained resources should have the feature of being lightweight to some extent, and should still offer some fundamental security protection.

12.3.2.1 Security Measures on Passwords

IoT devices are managed by human users, and user access to an IoT device usually needs to have an account, which includes user name (not always necessary) and a password (subject to user setup). There are weak passwords which are vulnerable to dictionary attack and brute force attack. Even worse, there are cases where the initial password set up by the product manufacture is used, and the worst case is that the factory passwords are not allowed to change at all. In order to reduce the chances of choosing weak passwords, or to use the factory default passwords, there need to have some security measures related to user passwords. This kind of security measures are easy to test.

***Measure*-18** User account should have a password.
This security measure seems to be so obvious. However, many IoT devices with embedded operating systems allowing only one user, and the designer/manufacturer may ignore the necessity of a password.
Rationale: A password is always necessary whenever human access to the system of an IoT device.
Test method: The test on this security measure is simply to check whether accessing to the IoT device needs a password. This is an existence examination test as described in Sect. 12.4.1.

***Measure*-19** IoT devices should allow users to change their passwords at their discretion.
It is a fundamental requirement for an information system to allow users to change their passwords at discretion. However, this is not always the case for IoT devices.
Some IoT devices are badly designed with respect to security protection. Even if they requires a password for user access, they may disallow users to change their passwords. Such cases are vulnerable to network intrusion.
Rationale: There have been cases where IoT devices do not allow users to change the factory passwords. This is why this security measure is made explicit.
Test method: It is trivial to test whether an IoT device allows users to change their passwords. This is a validity test as described in Sect. 12.4.5.

***Measure*-20** There should be requirements on the format of passwords with respect to the minimum length and the number of different types of characters, aiming at avoiding weak passwords.
This requirement comes from the fact that many weak passwords are being used. This fact is seen from published user passwords and login information. It is

seen that many users used weak passwords such as *88888888, 12345678, admin*, which are easy to guess.

This requirement on passwords has been widely applied to information systems, which seems to be effective to minimize the use of weak passwords. A typical case of such requirement is that the length of a password should be at least 8 characters, and the password should contain at least two types of characters, where the choices of the characters are small case of English alphabets, capital case of English alphabets, numbers, and other characters available from a normal keyboard.

Rationale: Although some IoT devices allow users to change passwords, it is not always a mandatory action. Moreover, the IoT devices may not check whether the new password is weak or not. Requiring the format of passwords may ensure the security level of passwords on average. For example, if a password is required to have at least 8 characters which contain at least two types of characters, then the password is likely to have good resistance against password guessing attack, although occasionally weak passwords still exist even following this requirement.

Test method: The test on this security measure is simple. It is a typical kind of validity test as described in Sect. 12.4.5.

***Measure*-21** There should be a mechanism to reset a forgotten password.

Passwords can be forgotten, and reset a forgotten password is hence needed. There are different approaches for this purpose. One approach is to adopt a physical switch on the IoT device for the purpose of password reset. This mechanism is not suitable for IoT devices installed in an open area or a place that is difficult to reach.

Another approach is to define a special command forcing password reset. In order to avoid the password from being changed by unauthorized users, there should be a security protection for the password reset. For example, if a special key is used for user authentication and security protection of the new password, then online password reset is possible. What if the user even forgets the special key used for password reset? It is a management problem instead of a security problem.

While password reset is convenient, this functionality might be attempted by attackers trying to illegally reset other users' passwords. This security threat can be minimized by some security mechanism in the process of password reset.

Rationale: It is a common phenomenon for a user to forget his/her password. Password reset for IoT devices should be made simple and easy to operate, while ensuring the security.

Test method: The test on this security measure can be accomplished by going through the password resetting process. This is a validity test as described in Sect. 12.4.5, and the test process should take human on-site examination test as described in Sect. 12.4.4.

12.3.2.2 Security Measures on Important Data in IoT Devices

***Measure*-22** Some key parameters for IoT devices should not allow users to change.

Many IoT devices are managed by operators, in such cases, users are not allowed to change important parameters such as seed keys. A typical example is when a mobile phone is used as an IoT device, or an IoT device uses mobile communication systems to transmit data, in this case, some parameters (e.g. the IMSI and the seed key) for secure mobile communications are fixed in a SIM card, where users are not able to change those parameters.

Rationale: Some key parameters for IoT devices such as MAC address, port numbers, may not allow users to change, particularly for IoT devices with constrained resources. This security measure is to limit the outcome when attackers intrude such devices and have full control over them.

Test method: The test process on this security measure needs to consider different cases. When the security parameters are fixed in an external hardware, such as a SIM card to a mobile phone, then it is treated as meeting the security measure If the security parameters are all managed by software, the test process should attempt to change the parameters which are supposed to be prohibited. This is a kind of attack simulation test as described in Sect. 12.4.6.

***Measure*-23** The data in secure sector for IoT devices or chips should not allow users to read.

The right of reading data is different from that of writing data. When users are restricted from creating or modifying data, it does not mean that the users are not allowed to read. When users are restricted from reading some data, it does not mean that the users are not allowed to write.

If the data is stored in the secure sector, it will be overwritten when software is updated. Users may be able to overwrite the whole software and the secure data, but may not be able to read the existing data.

Rationale: Sometimes the right to read certain part of data has more severe security risk than that to write it. For example, when the data in a secure sector is an encryption key, the right of reading the key means the capability of decrypting the encrypted data. However, the right of writing does not help, because when a key is overwritten, it simply makes the device not working properly.

The right of reading and that of writing are different, and cannot bind together, hence this security measure is made separate.

Test method: The test on this security measure makes sense only when there is some data stored in a secure sector. The most common such devices are security chips and RFID tags. By attempting to read the data in the secure sector directly and see whether this attempt is successful, the test result can be concluded. This test can be a kind of suitability examination test as described in Sect. 12.4.3, or a kind of attack simulation test as described in Sect. 12.4.6.

***Measure*-24** Users with the role of manager should be able to control the right of newly added users.

This is a common security measure for traditional information systems. If some IoT devices also have the functionality allowing the manager to create new user accounts, then the same security measure should apply.

Rationale: This security measure is to reduce the security risk caused by other user's account instead of the manager's account. Trojan horses may come to the system via a normal user's account, and restricted user right can also limit the risk by the Trojan horses and other kind of viruses from a user's account.

Test method: The test process is to go through the process of adding a new user, and make some limitation on the right of the newly added user, and verify whether the newly added users indeed have the right as expected. This is a validity test as described in Sect. 12.4.5.

***Measure*-25** Operating system parameters should not allow normal users to change.

In a traditional information system, a normal user is usually allowed to change some parameters of the operating system, particularly those configured for the user account. However, this functionality does not seem to have wide applications in IoT systems, instead, it may increase the security risk.

Rationale: When an operating system has a manager account and some other normal user accounts, most of the system parameters should not allow normal users to change. This is to reduce the risk of malicious users, or users' mistake operation.

Test method: The test on this security measure can be done by attempting to change system parameters from a normal user account. This test is a kind of attack simulation as described in Sect. 12.4.6.

12.3.2.3 Security Measures on Application Software

***Measure*-26** Developers of operating systems for IoT devices should have good business credit.

Traditional operating systems for large information systems are normally developed by large companies with good business reputation. For IoT devices, there are a variety of different operating systems, and there are a variety of developers as well. Large amount of the operating systems for IoT devices are developed by individual manufactures based on open Linux core. In order to ensure certain level of reliability and security, those developers should have established good business credit. This can be reflected by some certificates, authorizations, customer feedback, media reports, or white/black name list from relevant industrial associations.

Rationale: This security measure reminds product buyers not only check the functionalities of the IoT devices, but also pay some attention to the business credit of their manufacture.

Test method: This security measure does not need a standard test. The test is like a suitability examination test as described in Sect. 12.4.3. More specifically, As long as the business can provide a convincing evidence of showing its good

business credit, this security measure is treated as being satisfied. In some sense, this security measure is subject to the users' trust on the IoT device providers, which can be IoT device manufactures or IoT application system developers and operators.

It is noted that the business credit is a dynamic status, which varies from time to time. A business having good business credit at one time may loss this status in a later time, and vise verse. So, it makes sense to check the business credit at the time of purchasing the product from the business, and do not need to care about whether the credit status is changed later on. The business credit needs to be checked from time to time if the business is a service provider.

***Measure*-27** Important IoT application software should have passed some relevant security test made by a qualified third party.

Some IoT devices have application software installed to the operating system. Such application software is often designed for specific IoT applications. Considering that end users may not be able to examine the functionalities and properties of the application software, the application software should be formally tested by a third party who has qualification of software test.

Although this security measure does not guarantee the software to be safe and reliable, the process of third party test is targeted at reducing the risk of discretional software which may induce big security risks, or even having security trapdoors by the software itself.

Rationale: This security measure reminds users of IoT devices (particularly the endpoint user devices) to be careful when to use a third party application software. This has been well done practice. For example, many applications for iPhones are available from Mac Store, which have been tested by the MacStore platform. This test is helpful to iPhone users.

Test method: The test on this security measure is somehow like an existence examination test as described in Sect. 12.4.1. More precisely, it is to check whether there is a proof or permission made by a qualified third party.

***Measure*-28** IoT application software should have the functionality of online update.

IoT devices having application software usually have relatively good computing and communication resources. Application software may need to upgrade due to adding some functionalities, or to amending some security flaws. Therefore, software online update is a convenient and important feature, and has been a mature technique in traditional information systems.

Rationale: The necessity of online update of application software is obvious. It is to ensure the consistence in security and service provision.

Test method: This security measure can be tested only when there exists an update of a specific software. If this is the case, then the test can be done by going through the process of software online update and check whether the software update is successful. This test is a validity test as described in Sect. 12.4.5.

***Measure*-29** Some key parameters can only be accessed by internal functions.

This security measure is mainly about security chips, where the data in the chips cannot be accessed externally, and can be accessed internally via a pre-designed interface.
Some secure data such as encryption key is stored in an area which cannot be accessed externally but can be used by internal functions as has been so designed.
Rationale: This security measure is to avoid illegal access to confidential data, and does not conflict with the functionality that external devices can overwrite the data, which would overwrite all the data in the chip.
Test method: There are tools to access arbitrary sections of normal chips. Those tools do not suppose to work for security chips. The test on this security measure includes two aspects: (1) whether the target device is a security chip which has already gone through sophisticated security test, or the target IoT device is based on a security chip (2) whether there is a security flaw in the design of the internal data which is not supposed to be accessible externally.
The first test is like an existence examination test as described in Sect. 12.4.1, and the second test is like a suitability examination test as described in Sect. 12.4.3, it is subject to the test methods and perhaps with some manual analysis. This security measure is designed for devices with high security requirement.

12.3.2.4 Security Measures on Operating Systems

Operating systems for IoT devices may or may not be well known operating systems such as MS Windows, Solaris, Android, Linux, but may be closely related to one of them. There are many individual operating systems for IoT devices due to their diversity, hence security measures need to be made specific and explicit.

***Measure*-30** When a common commercial operating system is used, it should comply with some related security policies.
Security policies for IoT devices include the use of access control, intrusion detection, security auditing, system backup and recovery, software update.
Security policies are often made available by national or industry standards and regulations. Different industries may have different set of security policies, and different applications from a same industry may also have different set of security policies.
Rationale: This security measure is about a common security requirement, and the relevant techniques are quite mature.
Test method: The test on this security measure is simply to confirm items from a checking list of pre-defined security policies. This is a typical case of suitability examination test as described in Sect. 12.4.3.

12.3.3 Network Security Measures

As Fig. 2.1 shows, the Internet of Things has a network layer, which is meant for wide area networks. The most common such networks include the backbone network of Internet, mobile communication networks, and newly developed low power wide area networks (LPWAN).

Apart from the network layer, there are local networks in the perception layer of IoT. These networks are for the communication among the devices in the perception layer, including the communications between endpoint IoT devices and an IoT gate node. Since many IoT devices communicate via wireless networks, it is easy to capture their communication data, and to send forged data to any target device. If there is no security protection in the local network within the perception layer of IoT, then there will be large amount of attacks targeting at the perception layer of IoT.

This section lists some security measures about network security, including the devices in a local area network as well as the devices linking a wide area network.

12.3.3.1 Security Measures on Network Ports

***Measure*-31** Network ports being unnecessarily open to the network should remain closed.

An IoT device with communication capability usually has more than one network port, particularly those with IP connections to the Internet. Many network administrators only ensure to open the network ports in use, and ignore the states of those ports that may not be used. The network ports open for applications should be checked carefully in case of possible intrusions from attackers. However those network ports open to the network but not being used by applications may be used by attackers to intrude into the system. Therefore, close the network ports unnecessarily open to the network can reduce the chances of network intrusion.

Rationale: This security measure is specific to IoT devices, it is not treated as important in traditional information systems. The purpose of this security measure is to reduce the chances of being found by attackers in the process of network scanning, which is the first step in finding targets to attack/intrude.

Test method: The test on this security measure is to make sure which network ports are necessarily open, and check whether there are any other open network ports. It is a specific validity test as described in Sect. 12.4.5.

***Measure*-32** The network ports should keep closed during the time they are not used.

***Measure*-30** is about the unused network ports, this security measure is about the used ones. Many IoT applications require some endpoint IoT devices to transmit data periodically, with occasional exceptions. In those cases, the IoT devices do not need to keep being exposed to the network all the time.

However there may be some exceptional cases when the IoT devices are requested to send data, or to execute some command. In order to take into consideration the occasional exceptional cases, the endpoint IoT devices can open the network ports a bit more frequently than the scheduled time of data transmission, just to listen whether there is data/command sent to them. After a short time slot of listening, the network ports are closed again waiting for the next time to open.

One typical practical case is domestic water meter reading (same as gas/electricity meter reading). A few times meter reading a day is sufficient, so the meter reading device only needs to open its network port once in every few hours. However, in case of occasional data request from the management center, the meter reading device should open the network port more frequently than the period for data uploading. For example, the device opens the network in every minute, and keeps the network port open for 2 s. When the management center needs to send exceptional message to the reading devices, the management center should keep sending the message once in every second or more frequently, and keep sending the message for at least 60 s. Considering the possibility of data transmission failure, it should keep sending the message a few more seconds.

Rationale: In traditional computer networks, the network ports frequently or occasionally used usually keep open all the time. However, this case should be changed for IoT devices. The reasons include the following: (1) Many IoT devices only use the network occasionally, for example, a few time of data transmission, and more frequent time of network listening each last for a short period of time. (2) Most of the IoT devices do not have good security protection, and are vulnerable to network intrusion. Hence, a practical and effective security policy is to keep the devices invisible from the public network as much as possible.

Test method: The test on this security measure can be done by checking the status of the target network port, to see whether the network port is open for a small period of time, or to finish sending data, and then closed for a relatively long time. This is a kind of validity test as described in Sect. 12.4.5.

12.3.3.2 Security Measures on Identity Authentication

Identity authentication is a common security requirement. It is used in almost all the secure communications, particularly in a system with multiple users.

The purpose of identity authentication is to ensure that the identity of the remote entity (or the identity of the data source) is indeed what it claims to be, making masquerading a difficult task.

***Measure*-33** There should be a mechanism for identity authentication when an IoT device requests to connect to a network.

Identity authentication in IoT applications can help to eliminate the network connections from illegal devices. Identity authentication can be used for authenticate the identity of an IoT device, or the identity of a user using the device.

Rationale: Identity authentication is a common security requirement, which applies to IoT devices when the authentication process consumes acceptable amount of resources.

Test method: Since there are many means of identity authentication, some information about the authentication method is known before the test can be done.

In general, the test on this security measure is to check the availability of the claimed authentication services and process, so this is a kind of validity test as described in Sect. 12.4.5.

For example, if the authentication is for a user, then user account associated with a password and/or other means of authentication should be used. If the authentication is for an IoT device, then challenge-response authentication mechanism may be employed. Since the target authentication method is specified before a test can be done, this is also a kind of suitability examination test as described in Sect. 12.4.3.

A general but weak test on this security measure is to attempt to establish a connection with an unauthorized identity, and see whether there is a possibility of successful connection. In this sense, the test can be treated as a kind of attack simulation test as described in Sect. 12.4.6.

***Measure*-34** Sometimes the identity of an RFID tag should vary dynamically.

The identities for IoT devices are usually fixed all the time in their lifetime, and the identities are often unique in their applications. It would be better for the identities to be globally unique.

However, for RFID tags, sometimes their identities need to change dynamically, so as to protect their privacy. This security measure is mainly for RFID tags, and only in specific applications such as logistics and entrance access (e.g. RKE). This security requirement is called the identity privacy protection for RFID tags. Identity privacy protection is not a new technique. In mobile communications, the identity of a mobile user changes from initial IMSI to TMSI which then keeps changing from time to time, unless a new session of user authentication is lunched.

There are different methods to achieve identity change. One common method is used in the mobile communications, where a mapping from the real identity (e.g. IMSI) to the temporary identity (e.g. TMSI) is maintained by the server, and another common method is to use a probabilistic encryption algorithm to encrypt the real identity which yields a different ciphertext every time the encryption is performed.

Rationale: The dynamic change of the identity of an RFID tag is meant to protect illegal readers from being able to tell whether an RFID tag at a specific time and location is the same as the one in a different specific time and location [11], which would give some tracing information.

Test method: The test on this security measure is to check when the condition for the identity to change meets (e.g. when it is being read by an RFID reader, or when some time passed (in case of time based identity change), whether the identity really changes. This is a validity test as described in Sect. 12.4.5.

12.3.3.3 Security Measures on IoT Gate Nodes

IoT gate nodes are not the same as the network gates in Internet or other networks. IoT gate nodes are like sinks in wireless sensor networks who collect data from the endpoint sensor nodes and send the data to a management center, and forward commands from the management center (when the management center incorporates the functionality of authentication center) to the target endpoint sensor nodes.

***Measure*-35** The IoT gate nodes should be able to process data fusion.

An IoT gate node is such a device/machine that collects data from a few endpoint sensor nodes and then forward the data to a management center, and forward messages/commands from the management center (e.g., coming from a user) to the endpoint sensor nodes. Since the size of sensing data may be small, in order to improve the efficiency of data transmission, the IoT gate nodes may need to integrate the data from different endpoint sensor nodes and then forward to the management center. The process of data integration needs to be able to handle different kinds of data, and the technique is called data fusion.

There are many mature techniques for data fusion. An intuitive approach is to pack the data from different endpoint sensor nodes together with the corresponding data header, sensor ID, etc., and the receiving management center then is able to separate the data and handle them properly.

Rationale: This security measure is for the IoT gate nodes serving different kinds of endpoint sensor nodes. Most IoT gate nodes have the ability to serve different kinds of sensor nodes, and have relatively large computational capability.

Test method: The test on this security measure needs to capture a data packet from the target IoT gate node, and check whether the data packet contains different data from different endpoint sensor nodes, which may be of different data type. This is a validity test as described in Sect. 12.4.5.

***Measure*-36** The IoT gate nodes should be able to limit a maximum number of connections.

The technique for this security measure is simple, it can be done by checking the number of connections, and when the number reaches an upper bound, i.e., the maximum number of allowed connections, no more connection request is processed. Occasionally the number of connections may exceed the limit, and in such case, no more connections are allowed.

Rationale: This security measure is to prevent possible DoS attack and DDoS attack. The determination of the maximum number of connections is based on a number of factors, including the capability of the IoT gate node and the characteristics of the application.

Test method: The test on this security measure is trivial: simply increase the number of connection requests, and see whether the number of connection can exceed the upper bound. This is a kind of validity test as described in Sect. 12.4.5.

***Measure*-37** The IoT gate node should be able to handle security services between the gate node and the central database in upper layer, and the security services between the gate node to the endpoint sensor nodes.

It is known that the central database of an IoT system is located in the processing layer. When the database connects to the endpoint IoT devices, the connection is done via the IoT gate node. In the case of direct connection, the endpoint IoT device is actually a combination of an endpoint IoT device and a gate node.
Rationale: The connection between an IoT gate node to endpoint IoT devices is often wireless connection, while the connection between the gate node and the central database is usually wired (e.g., the Internet). The wireless connection often has not security protection except the password based wireless connection (e.g., WiFi connection), while the wired connection should have better security protection, since the transmitted application data and command data should not be intercepted, tempered, or the identity being spoofed.
The need for the security separation is because the IoT gate nodes often have much more computing power than some resource-constrained endpoint IoT devices.
Test method: The test on this security measure is to check whether the security functionalities in the wired connection and the wireless connection are different. Even if the security algorithms are the same, encryption keys should be different. This is a special case of validity test as described in Sect. 12.4.5.

12.3.4 Data Security Measures

Data plays the key role in all communications and networks, as well as information systems. The purpose of information security is essentially to protect the data and its processing platform (e.g., information systems). However, there are security techniques targeting at protecting the data itself, while others provide a variety of other security services.

Data security can be implemented in different processes. It include the processes of data storage and data communications. The data security protection techniques are mainly cryptographic methods, which have been proved to be effective.

12.3.4.1 Security Measures on Data Storage

***Measure*-38** Confidential data should not be stored in plaintext format.
This security measure is for normal data storage which could be accessed by attackers. Storage of confidential data such as user passwords in plaintext format obviously has high security vulnerability, because such plaintext data can be accessed by any user who is able to access the data.
Rationale: This security measure seems to be a common sense. Unfortunately, we often see this phenomenon in reality. This security measure reminds data service providers to store confidential data in a secure manner.
Test method: The test on this security measure can be done by checking the format of confidential data (assuming the knowledge about what confidential data

is) in storage. This is a specific case of suitability examination test as described in Sect. 12.4.3.

To check whether a message is in the form of ciphertext is not a simple task. If the encryption key is given, then the test is a simple correctness examination test as described in Sect. 12.4.2. Otherwise, it is difficult to judge whether a message is a ciphertext or not.

***Measure*-39** Storage of important data should have integrity protection.

The purpose of data integrity protection is to detect possible change of data during transmission, which can be made by attackers or caused by hardware failure. There are different techniques to provide data integrity protection. The cryptographic message authentication code (MAC) is a common method to provide data integrity protection, but may not always be suitable for IoT applications when the data size is small. This security measure is not restricted to which method used, it only requires the availability of the data integrity protection service. In practical applications, the method providing the data integrity protection service needs to comply with the industry requirements, if there are relevant such requirements.

Rationale: The requirement of data integrity checking has become less important than before since the computer hardware is quite reliable nowadays. However, data may be tempered or destroyed in networked environment, particularly in the case of distributed data storage. Hence the integrity protection in data storage is important.

Test method: It is easy to check whether data tempering can be detected. However, it is difficult to check whether the method providing the data integrity service has sufficient security. This test method combines the attack simulation test as described in Sect. 12.4.6 and suitability examination test as described in Sect. 12.4.3.

***Measure*-40** A secure IoT device should have a secure storage section.

A secure IoT device is for import applications, where endpoint sensor nodes should have high security protection. Since endpoint sensor nodes are easy to capture by attackers, if an attacker can get the encryption key of a sensor node, then it easy to forge data. If the whole perception layer uses a same encryption key, then getting the encryption key means full control of the perception layer, which may result in severe security problems.

Secure storage section can be found in many security chips, so by employing any of such chip, this security measure can be met.

Rationale: This security measure is mainly for the applications requiring secure chips for data security protection. However, other means of providing the same service may also be acceptable. For example, storage of data into a removable device is such an alternative.

Test method: If the target device uses a security chip with a secure storage section, then this security measure is treated as being met. Other means of providing this security requirement should provide convincing proof of being reasonable, and should be tested accordingly. This test is a kind of suitability test as described in Sect. 12.4.3.

12.3.4.2 Security Measures on Data Communication

***Measure*-41** Confidential data should be encrypted in its transmission over publicly accessible channels.

Data encryption provides data confidentiality service, which suits confidential data protection in its transmission. Data encryption is a mature technique and has wide applications to many IoT systems as well as traditional information systems.

Rationale: Encryption algorithms have proved to be effective means to protect data confidentiality. Encryption of confidential data in transmission has been a common practice.

Test method: The test on this security measure is to check the existence of the data encryption during transmission. This is an existence examination test as described in Sect. 12.4.1. Since the existence examination test on ciphertexts is difficult to conduct, practically the test on this security measure can be done by correctness examination test as described in Sect. 12.4.2.

***Measure*-42** Important data may need to be encrypted with an authorized encryption algorithm in its transmission.

Different kinds of data transmission should choose different kinds of encryption algorithms. For example, if the data is transmitted over the reliable wired network, then a block cipher may be appropriate. If the data is transmitted over the wireless network, and the data has high low tolerance for time delay, then the mechanism of byte-based encryption algorithms (e.g. stream cipher or block cipher in CTR mode) might be appropriate.

An authorized encryption algorithm can be a national standard algorithm, or an international standard algorithm, or a commercial standard algorithm. In some specific applications or special industry, only authorized encryption algorithms are allowed to use.

Rationale: Sometimes the use of encryption algorithm has to comply with industrial or national standards and regulations, in this case, the encryption algorithm for use should be an authorized one.

Test method: This is a typical correctness examination test as described in Sect. 12.4.2. The process of test needs to get some transmitted data as test sample, and to get the corresponding plaintext. It is assumed that the encryption algorithm and the encryption key are provided to the test.

***Measure*-43** Command data in transmission should have integrity protection.

In IoT applications, a command data is a small piece of data, which usually does not need confidentiality protection, because the command data usually has small varieties, i.e., only a small number of commands are defined in a system. For example, a command of turning a switch into "on" or "off" status has only two possible values, hence its confidentiality makes little sense. However, the integrity protection is essential, because any malicious data tempering, if cannot be detected, may cause a severe damage.

Traditional methods of data integrity protection include message authentication code (MAC) and digital signature. However, they may not be the suitable choices

for IoT application data. There are other reasonable means of data integrity protection methods with light-weight feature. For example, by encrypting the data together with the node identity information, services such as data confidentiality, data integrity protection, and identity authentication can all be provided at the same time. Such an example can be found in Sect. 12.6. The method of using an encryption algorithm to provide data integrity service as well as other services is a lightweight and practical approach in IoT applications.

Rationale: Data integrity protection is meant to thwart data tempering, because tempering on command data is a severe security threat. This is why data integrity protection is important.

Test method: Like the test on data integration protection for data storage, the only difference is that the test data should be captured from its transmission.

If the key in the data integrity protection algorithm (e.g. an encryption algorithm, or a MAC algorithm, or a digital signature algorithm) used to verify the correctness of the data is available, then the test is a correctness test as described in Sect. 12.4.2. If the key used to verify the correctness of the data is unknown, then the most practical test method is probably attack simulation test as described in Sect. 12.4.6.

***Measure*-44** In some cases, integrity protection on transmitted data may need to use an authorized algorithm.

Standard data integrity protection algorithms are the most commonly used algorithms. Such algorithms include cryptographic message authentication code (MAC) such as MD5 [9], SHA1 and SHA2 [3]. MD5 is not recommended to use due to its security weakness. However, message authentication codes are not the only choices, it may be possible to use a standard encryption algorithm to provide message integrity service, this is a practical approach for resource-constrained devices in IoT environment.

Rationale: In industry, security services and technologies may have to comply with industrial and/or national standards and regulations. This is the case when the standards or regulations apply to the case of data integrity protection.

Test method: The test on this security measure is a typical case of correctness examination test as described in Sect. 12.4.2. In such a correctness examination test, knowledge such as the data format, the algorithm, and how the algorithm is used, should be known in advance before the test can be done. Given the test data together with its integrity protection tag, the integrity protection algorithm, and the key used by the algorithm, it only needs to perform the integrity algorithm and compare the outcome of the algorithm with the captured data integrity tag.

***Measure*-45** There should be a mechanism to resist replay attack on transmitted data.

Replay attack is to re-send a captured data to the destination receiver at a later time, hopping the data to be accepted (not just received) by the receiver. Regardless whether or not the data has security protection such as being encrypted or having integrity protection, the replayed data can pass all the security checking. However, the data is re-sent at a wrong time, hence may cause confusion or harm to the system.

Rationale: not possible for resource-constrained devices to record large amount of historical commands (although small amount of such data is possible). On the other hand, simple commands like "open/on" and "close/off" may correspond to two ciphertexts. If the encryption algorithm is not probabilistic, in such case, recording the historical data and avoiding historical data to be replayed may result in normal command data to be wrongly rejected.

In IoT practical applications, the technique of resistance against replay attack is to employ a counter or a timestamp, which indicates the freshness of a message. There are advantages and disadvantages of using a counter or using a timestamp. Which method is more convenient depending on the real application and available execution environment.

When a counter is used to mark the data freshness, the counter value is incremented (usually increment by 1) each time a message is transmitted. When the target receiver receives the data, the counter value is checked and compared with the locally recorded value. If the received counter value is larger than the locally recorded one, then accept the data, and update the local counter value by the received one. Otherwise the data is rejected.

The advantage of using a counter to provide data freshness service is that, the probability for any replay attack to succeed is negligibly small, while its disadvantage is that, for any two-party communication, both the sender and the receiver need to maintain at least one counter for one-way communication.

When a timestamp is used to mark the data freshness, both the sender and the receiver need to have a clock. The sender attach a timestamp to the data, and the receiver checks the timestamp attached to the data, and compare it with the local time, if the difference is within an allowable range, then the data is accepted otherwise the data is rejected. The size of allowable time difference is defined in advance, which is determined by the specific application, which may differ significantly. For example, some systems may allow the difference to be a few seconds, while other systems may allow a few hours.

The advantage of using a timestamp to provide data freshness service is its convenience. The disadvantage of using a timestamp includes the following:

1. Both the sender and the receiver should have their clocks properly synchronized. If the sender's clock goes faster, a data from the sender may arrive the receiver at an earlier time (earlier than the local time of the receiver) may be rejected If the sender's clock goes slower, then the allowed time for the data to arrive the receiver is smaller, which increase the chances of legitimate data to be rejected.
2. The allowable time difference should be set appropriately. If the allowable time difference is too large, then within the allowable time difference, replay attack is possible if the allowable time difference is too small, then the chances for legitimate data to be "timeout" is large, hence is wrongly rejected, which reduces the reliability of successful communication.

The technique of using a counter to provide data freshness is suitable for one-to-one communication environment, and the approach of using a timestamp is suitable for group communication environment, e.g. broadcast communication.

To unify the two methods, we call the data tag (timestamp or the counter) as *data freshness tag*. The data freshness tag is attached to the data when the data is transmitted.

Replay attack may not be a severe security threat for the Internet environment, however, it is an effective attack in IoT environment, since in many cases, an IoT application data can be carried by a single datagram. Replaying such a datagram may result in a wrong message to be processed. If the data is a command, then execution of the command in a wrong time may cause severe damage. For example, if a command to open a switch is replayed when it is supposed to be off, or the other way round, then what a damage it may cause depends on what the switch is used for.

Test method: The test on this security measure depends on the mechanism of replay resistance. A general test method on the resistance of replay attack is to use the attack simulation test as described in Sect. 12.4.6.

***Measure*-46** There should be a mechanism to resist active replay attack on data in transmission.

Here active replay attack means that the attacker may temper the captured data then re-send the tempered data instead of the original data. A common tempering by an attacker on transmitted data with freshness protection is to change the data freshness tag. In order to protect the freshness tag from being tempered, the data freshness tag should be protected by a cryptographic function such as an encryption algorithm, or a message authentication code, or a digital signature algorithm.

It is noted that this security measure is a higher level requirement than ***Measure*-45**. If this security measure is met, so is ***Measure*-45**.

Rationale: Simply adding a freshness tag to a message is not enough to resist against active replay attack. Appropriate security protection of the message freshness tag is necessary. When the message freshness tag has security protection, then any tempering on the message can be detected, because any modification of the message would result in the freshness tag to be invalid, and the exception is of negligible probability.

Test method: Activity replay attack is an improved measure on replay attack. The test on this security measure is based on the replay attack, and attack simulation test also applies to this test. It should be noted that, when doing the attack simulation test, the process involves different tricks on the attack simulation, including changing the replay waiting time, tempering the message, and tempering the message freshness tag.

***Measure*-47** In some cases, traffic flow protection needs to be provided to thwart traffic flow analysis.

Network traffic flow analysis is a common security threat in traditional networks, it can be a more severe security threat in IoT environments. In an IoT system, traffic flow analysis may reveal some confidential information. For example,

assume that the data about residential water consumption is encrypted for security reasons. If someone noticed that the encrypted data from a specific household changes to be small, it means that the household consumes little (or even none) water, which may mean that the household has not been occupied. This information reveals some private information about the occupants of the household, and may even risk the safety about the household.

By traffic flow protection, it means that the size of the ciphertext looks similar regardless of the plaintext size. The objective of this security measure is to make it hard to guess the plaintext size by observing the size of a ciphertext. There are many mature methods to balance the ciphertext size.

Rationale: The purpose of traffic flow protection is to eliminate the privacy leaking by the size of the encrypted message. Some specific size of message (e.g. small size) may reveal some information about the message, although it does not necessarily reveal the content of the message.

Test method: The test on this security measure is a validity test as described in Sect. 12.4.5. It is simply to check whether the ciphertext sizes have obvious difference when the plaintext sizes differ.

12.3.4.3 Security Measures on Data Backups

***Measure*-48** Important data should have a backup.

In IoT systems, application data is often sensitive with time, so the backup of such data has little demand. However, configuration data for IoT devices is not time sensitive and is important. When an IoT device is out of order and needs to be replaced, the new replacement should be configured properly. The configuration data of the original device hence becomes important. When extra devices need to be added into an IoT system, they may have same or similar configuration with neighboring devices. Hence, the configuration data of IoT devices should have a backup, since this kind of data is often used.

Practically, configuration data backup of an IoT device can be kept in the data processing center that serves the IoT devices.

Rationale: The data backup service on configuration data is necessary, particularly the configuration data of some IoT devices that need recovery or replacement when being out of order.

Sometimes data backup is necessary for some important temporary data and application data.

Test method: The test on this security measure is a typical existence examination test as described in Sect. 12.4.1, it is simply to check the existence of backups of the configuration data for IoT devices.

***Measure*-49** There should be a mechanism for fast data recovery with data backups.

When configuration data backup is used to recover an IoT device after the device goes disorder, the process is similar to the configuration process of a backup device, which is simply to write the configuration data into the device.

However, in some cases, writing data into a device needs to pass authentication and verification process, which can be implemented by using an encryption algorithm, or a digital signature algorithm. In both the cases, a key is needed by the algorithm. If the key is lost, then data backup becomes more complicated. Usually the backup process is to re-install the backup data to the device from an early backup data.

The pre-condition for data recovery in IoT is the existence of data backups.

Rationale: Important IoT devices in industrial IoT are master control stations or the kind. Such devices should have fast recovery.

Test method: The test process on this security measure is to go through the process of data recovery. The test process is a validity test as described in Sect. 12.4.5. It is noted that, how fast the process behaviors depends on the specific applications and requirements by relevant industries, hence this is also a suitability examination test as described in Sect. 12.4.3.

***Measure*-50** Some key data should have a dynamic backup made on-line.

On-line data backup is not supported by all the IoT devices, and is not needed by all the IoT applications. This security measure is for the cases when dynamic data (e.g. a counter value which changes from time to time, or a session key) needs to make a backup, which should be done on-line, because off-line backup of data from an IoT device is inconvenient and less practical.

The cost for on-line data backup is the consumption of electric power and communication bandwidth.

Rationale: A specific case of this security measure is machine-to-machine (M2M) communication, where data backup is kept for a while in case the data is frequently used, in such case, the data backup can be provided without needing to get the data from its original source.

Test method: The test on this security measure is to check whether the target data (which has specifications in advance) can indeed make a backup on-line. This is a special case of existence examination test as described in Sect. 12.4.1.

***Measure*-51** There should be a security protection for the process of on-line backup of important data.

Off-line data backup may not need security protection, because the backup process is often conducted in a trusted environment, and the data cannot be forged. For on-line data backup, the data may be forged, or tempered. So, there should be security protection on the data backup to avoid data forgery and data tempering. Sometimes data confidentially is also required.

Rationale: Security protection of the backup data is mainly to avoid data tempering, sometimes also to protect the content of the backup data from illegal access.

Test method: The test on this security measure is to check whether the on-line backup process has the specified security protection. This test is a kind of suitability examination test as described in Sect. 12.4.3.

It should be noted that, the keys used by the security protection on the backup data should be independent of the backup data. The keys will be kept for use when the backup data is recovered.

12.3.5 Key Management Security Measures

In network environment, security services can hardly be reliably provided without the establishment of a trust. The trust in network environment means that one entity believes that the other entity knows some secret information which cannot be known by others.

For example, assume that there is a challenger A and a prover B. B has some secret information that A may or may not know, and no one else knows that secret. A can verify whether B knows the secret. Once A confirms that B really knows the secret information, it means that A has established the trust on B. There should be some assumption on that A has the ability to verify whether B knows the secret, this assumption is reasonable which can be established as the initial trust.

Trust is the bases for key management, and key management is the security bases for data encryption, data integrity protection, and identity authentication. In practical implementations, identity authentication is often integrated with key management.

***Measure*-52** An IoT system should establish an initial trust.

In communication systems, an initial trust is either a mutually shared key, or a public key. A mutually shared key can be trusted if the key was initially shared off-line, or established by the involvement of a trusted third party. A public key can be trusted by the mechanism of a public key certificate. However the process of verifying the validity of a public key certificate needs the public key of the CA (certificate issuing authority) for the certificate, which is often written into the IoT device by the manufacturer in the format of self-certificate of the CA.

Rationale: Without trust, authentication cannot be done with confidence, and hence cannot tell the truth or fake. Trust can be transferred, but cannot be created from none. So, an initial trust is necessary to support the security of the whole IoT system.

Test method: The test on this security measure is an existence examination test as described in Sect. 12.4.1. It is simply to check whether the IoT device has a secret key or a public key certificate (e.g., a self-certificate of some CA) written into the device (usually done by the manufacturer). Even if the key is as simple as "123456", it really makes sense.

***Measure*-53** Sometimes multi-factor authentication mechanism is needed for user authentication.

Multi-factor authentication means that more than one way is needed to conduct the user authentication. For example, a user needs to provide account information as well as a corresponding password, which forms one factor. In multi-factor authentication, the user needs to provide one or more extra authentication factors to complete the authentication process. For example, the user's fingerprint or a USB-key as an additional authentication factors, as so required by the system.

Rationale: Although multi-factor authentication (two-factor authentication in particular) has commonly been used in traditional information systems, this

mechanism has not yet become popular in the IoT systems. However, in some critical applications where strict user authentication is needed, the mechanism of multi-factor authentication is probably a good choice.

Test method: The test on this security measure can be done by attack simulation test as described in Sect. 12.4.6.

***Measure*-54** Session keys should be used to eliminate the frequent use of the seed key.

When large amount of data encryption is performed, or frequent data encryption is performed, if a same key is used all the time, it would increase the chance of the key being leaked [10]. A popular and effective method is to use the seed key to encrypt a randomly generated session key, and use the session key to encrypt the application data. The session key is valid only in one communication session. In case of public key encryption, the use of session key is to increase the efficiency of data encryption and decryption.

Rationale: The use of session key to encrypt application data has widely been used in traditional information systems. In IoT applications where the application data is not so small (e.g. video surveillance), it would be a good practice to use session key for data encryption. The use of session keys has been a mature technique and does not consume much computation resource.

Test method: The test on this security measure is a kind of suitability examination test as described in Sect. 12.4.3. More precisely, it only needs to check the following: (1) whether or not a session key is randomly created and used to encrypt application data (2) whether or not the session key is valid only in one session.

Although the test process is simple in theory, in practical test, one needs to have the seed key (or the private key of the receiver, in the case of public key encryption) as well as the application data for test.

12.3.6 Availability Related Security Measures

Availability is a special security measure, it is difficult to give a precise definition about what the availability is. In general, the availability is usually a set of coarse-grained security measures that are time-consistent.

The security test on the availability is different from that of other security measures. The availability should be a stable feature with time. Therefore, the availability should be treated as additional consideration in both IoT system construction and security test in a long run.

***Measure*-55** The data transfer protocol should be able to handle abnormal data and behavior.

When the data is not in expected format or size, this is an abnormal case, and should be handled properly. When data transfer protocol goes into an abnormal situation, e.g. when waiting for a response and nothing happens, in this case,

there should be a mechanism such as "timeout" to handle the abnormal case. This mechanism seems to be simple for traditional network communications, and may get neglected in IoT applications.

Rationale: Exception handling mechanism is always necessary, regardless whether it is to handle the abnormal data format, or abnormal protocol execution. This security measure is simply a transplant from traditional information system to IoT applications.

Test method: The test on this security measure is to check whether the methods to handle abnormal cases are suitable. It is a suitability examination test as described in Sect. 12.4.3.

***Measure*-56** IoT devices should have acceptable stability.

The stability of IoT devices is measured by how long time the devices work properly without problems (or equivalently, how often the devices work abnormally). Although occasional and temporary malfunction of individual IoT devices may not affect the functionality of the whole system, it reflects the stability of the IoT devices. This measure requires that the chances for an IoT device to have problems should be less than how the IoT device is designed.

Factors affect the stability of an IoT device include internal ones, such as hardware failure and software errors, and external ones, such as environments and unexpected events. Temporary malfunction and small errors are acceptable, but frequent occurrence of severe problems is a reflection of low stability and will affect the usability.

Rationale: The stability is an important security requirement (associated with availability service), and it is also difficult to have a precise measure. The stability of IoT devices is measured by a probability, measuring the chances of malfunction. It is not an easy task to measure the stability, and even to judge whether a problem is temporary malfunction is a question. So the acceptable stability of IoT devices often comes from the industry goodwill of the device manufacturer.

Test method: The test on the stability is not a simple process, and cannot be performed on a single IoT device. The stability of IoT devices is a measure reflecting the product quality, it can be reflected by customer feedback, and cannot be measured rapidly.

***Measure*-57** An IoT system should have high reliability.

Reliability of an IoT system is different from the stability of the IoT devices in the system. The reliability of an IoT system is measured by the chances that the IoT system stops working properly, and how severe the problems are, i.e., how much effort and cost are spent on fixing up the problems.

It is difficult to measure the reliability precisely. The reliability of an IoT system can be measured by the cost (financial cost and manpower cost) to making the IoT system come to normal from malfunction, and the lost caused by the malfunction of the system.

It is noted that the reliability problem of individual IoT devices may not severely affect the stability of an IoT system. An IoT system with many low stable devices may have good reliability, if the system is designed properly.

Rationale: The reliability of an IoT system is always a desirable security measure. It does not simply depend on the stability of the IoT devices that are used in the IoT system, it also depends on how the system is designed.

Test method: The test on the reliability of an IoT system is a long process. However, the designed reliability of an IoT system can be examined, which is similar to the suitability examination test as described in Sect. 12.4.3.

***Measure*-58** An IoT system should have good robustness.

The robustness of an IoT system is measured by the redundancy of IoT devices and the connectivity of the networking architecture. It measures the tolerance of system with dead devices and dead links, i.e., when some of the IoT devices stop working, and/or some network links become no longer available, the IoT system still works with acceptable quality.

Rationale: The robustness of an IoT system is a measure on the survival capability of the system when the system is under attack, and the attack may damage some IoT devices and break some communication links.

It is obvious that an IoT system with good robustness also has good reliability. However an IoT system with good reliability may not have a good robustness. For example, when an IoT system with stable devices can have good reliability, but may not have much tolerance on the malfunction of IoT devices. In practical applications, the robustness is a measure to be considered in design of the IoT system architecture.

Test method: The test on this security measure is similar to the suitability examination test as described in Sect. 12.4.3 to some extent, and is also similar to the attack simulation test as described in Sect. 12.4.6 to some extent. The difference here is that, no precise "yes/no" conclusion on the rest result can be properly made. The test may include examination on the maximum number of IoT devices allowed to stop working without causing the IoT system down, or the maximum number of links allowed to break down while the IoT system is still alive.

***Measure*-59** An IoT system should have certain level of intrusion tolerance in terms of OT security.

Intrusion tolerance is a measure about how robust a system is after illegal intrusion. If the intruder is a Ransomware which attempts to steal secret information or to make damages to the system, then IoT devices cannot provide intrusion tolerance. There are some studies on intrusion tolerance techniques for traditional information systems [2, 12], which usually require the system to have high capability of computation, or in a networked manner with some collaboration. Those techniques do not apply to the IoT devices with constrained resources. However, intrusion tolerance can be provided by a special architecture, provided that the intruder does not damage the system where they accessed, but try to control over other terminal devices linked to the system. This is the feature of OT security, and is particularly the case when the terminal devices are industry machines. In IoT systems, illegal and malicious control over other terminals having network connection with the machine being intruded is also a

potential security threat, this kind of threat is more hidden and is mostly targeted at control systems.

Rationale: Intrusion tolerance is an important security measure for Iot devices, particularly for devices in IIoT systems. By a properly designed security architecture, it is possible for an IoT system with resource-constrained IoT devices to have some capability of intrusion tolerance.

Test method: The test on this security measure can be done by attack simulation test as described in Sect. 12.4.6.

***Measure*-60** The lifetime of an IoT device should comply with the designed lifetime.

There should be a designed lifetime T and an allowable proportion Γ ($0 < \Gamma < 1$, usually measured by percentage) of exception, and it is expected that the probability for the lifetime of an IoT device to reach T is no less than $1 - \Gamma$.

Rationale: This security measure is a natural requirement which is usually made by industry and ensured by manufactures.

Test method: The test on this security measure is like a suitability examination test as described in Sect. 12.4.3. If the proportion of the same kind of IoT devices to reach the lifetime T is less than $1 - \Gamma$, then the security measure is no met. Note that different set of test samples may yield different results. The larger the test sample, the more reliable the test result will be.

***Measure*-61** There should be a mechanism to quickly replace the important IoT devices when they are not working properly.

Important IoT devices are those that play an important role not to be missed. For example, an IoT gate node is usually treated as such an important device, because an IoT gate node looks after a number of terminal IoT devices. If the IoT gate node stops working, then all the terminal IoT devices under the IoT gate node are out of reach, if there is no redundancy in the system.

Rationale: When an important IoT device is out of order, it should be replaced. However, the existence of a backup device is not enough. The time needed to fulfill the replacement process is also an important factor to consider, and this is the rationale of this security measure.

Test method: The test on this security measure can be done by checking the whole process of device replacement, and to judge whether the time used to complete the process is satisfactory. It is a typical case of suitability examination test as described in Sect. 12.4.3.

***Measure*-62** A replaced IoT device should have all the key configurations and functionalities the same as the one to be replaced.

The replacement of an IoT device is usually expected to work as the one before in terms of functionality. The new replacement can have more prompt reaction (e.g. with more computational power). Therefore, the device working as a replacement should be configured properly, with the same or similar configuration as the one to be replaced. In this case, backup of configuration data is a pre-condition for this security measure to be practical.

Rationale: This is particularly the case when the device is equipped with some security functionalities. Security functions usually need a key, and often need

an initial vector as well, sometimes more parameters are needed. Some of the parameters are confidential data (e.g. an encryption key), while others are less confidential (e.g. an initial vector, or a counter), but cannot be missed. The replacement may even need to have the same identity as the original device.
Test method: The test on this security measure is to check whether all the necessary functionalities are available and as before after device replacement. This is a special case of validity test as described in Sect. 12.4.5. It should be noted that, if a function is upgraded, this is treated as having the same functionality.

12.4 Security Test Methods in General

Security test methods are used to examine whether or not the claimed security functions and security services are really as what they are supposed to be. However, in some cases, it is difficult to guarantee that the test result can correctly reflect the real case.

There are different test methods, which meet different security goals. The following are the most commonly used test methods of the security measures.

12.4.1 Existence Examination Test

The existence examination test is to verify the existence of the claimed security measure. It can be done from the angle of security functionality, i.e., to test the existence of such a functionality, or from the angle of security service, i.e., to test whether the claimed security service is really made available.

Note that the security functionality is different from the security service. A same security functionality may provide different security services, while different security functionalities may provide a same security service. This is further described in Sect. 12.5.

It can be seen that, many of the security measures described above need existence examination test. For example, ***Measure*-12**, ***Measure*-13**, ***Measure*-14**, ***Measure*-18**, ***Measure*-27**, ***Measure*-29**, ***Measure*-41**, ***Measure*-48**, ***Measure*-50**, ***Measure*-52**, are specific cases where the existence examination test is needed.

It is noted that, the name of existence test seem to be explicit and could be done deterministically, however it is not the case. The existence test on cryptographic functions (e.g. an encryption algorithm) can be done provided that sufficient amount of test data is provided. The existence test on an output of a cryptographic function, e.g., a ciphertext, however, can hardly be done properly, because it is difficult to tell whether a piece of data is a ciphertext or not. When sufficient amount of test data is given, the test can be done by checking the statistical property of the test data, because ciphertexts usually have good randomness. Such a test is probabilistic, with some false positive rate and false negative rate.

12.4.2 Correctness Examination Test

Obviously, the existence examination test is not sufficient, particularly when the existence test only gives a certainly level of confidentiality about the correctness of the test result. A better test method is correctness examination test, where the correctness of the security function is verified. In most of the cases, when a security function involves a secret key, the correctness examination test needs to have the key available, otherwise the test cannot be conducted.

As a specific example, the correctness examination test on an encryption algorithm is to check whether the test ciphertexts are indeed encrypted by the specific encryption algorithm (with an encryption key being provided), it is obvious that the test data also includes the corresponding plaintext.

It can be seen that, the correctness examination test applies to many of the security measures described above. For example, ***Measure*-14**, ***Measure*-42**, ***Measure*-43**, ***Measure*-44**, are specific cases where the correctness examination test is needed.

12.4.3 Suitability Examination Test

Some security measures are not a simple "yes/no" option, their extent level is also considered. In such cases, the existence test and even the correctness examination test are not appropriate.

The suitability examination test is a coarse-grained test, usually more measure about the acceptable extent has to be determined before the test can be practically possible.

The security measures described above where suitability examination test is suitable include ***Measure*-16**, ***Measure*-17**, ***Measure*-23**, ***Measure*-26**, ***Measure*-29**, ***Measure*-30**, ***Measure*-33**, ***Measure*-38**, ***Measure*-39**, ***Measure*-40**, ***Measure*-49**, ***Measure*-51**, ***Measure*-54**, ***Measure*-55**, ***Measure*-57**, ***Measure*-58**, ***Measure*-60**, ***Measure*-61**.

It is noted that the suitability examination test usually should take human on-site examination test.

12.4.4 Human On-site Examination Test

Human examination is an important part in security test. Although sometimes such a test has little to do with technical methods, it is worth mentioning here.

Human on-site examination test is the personal examination about the physical devices, physical environment, related documents and other proof, to check whether or not the claimed security measure is provided and effective.

Although human on-site examination test in most of the cases is the presence of the examiners on the site. With the assistance of high-tech tools, some human-on-site examination test does not need the examiners to be physically present at the place of the IoT devices and installation environment. It can be done by other communication tools such as video conferencing, video monitoring, digital photos, or unmanned aerial vehicle patrol, as long as there is a security mechanism to prevent spoofing identities.

In general, human on-site examination is simply to check the compliance of the physical site with the claimed security measure. Many of the security measures listed in Sect. 12.3 should take human on-site examination test as part of the security test. The security measures described above where human on-site examination test is appropriate include ***Measure*-1**, ***Measure*-4**, ***Measure*-5**, ***Measure*-7**, ***Measure*-8**, ***Measure*-9**, ***Measure*-10**, ***Measure*-11**, ***Measure*-12**, ***Measure*-13**, ***Measure*-21**.

12.4.5 Validity Test

The test on a large number of security measures can be done by going through the process where the security measure is supposed to meet. This kind of test is called validity test. For example, ***Measure*-19** says that *IoT devices should allow users to change their passwords at their discretion*, the validity test is to check whether the target device/system really meets this security requirement. The actual validity test varies depending on the actual security measure, and there may be other skills needed in conducting the validity test.

The security measures described above where validity test is needed include ***Measure*-15**, ***Measure*-19**, ***Measure*-20** ***Measure*-21**, ***Measure*-24**, ***Measure*-28**, ***Measure*-31**, ***Measure*-32**, ***Measure*-33**, ***Measure*-34**, ***Measure*-35**, ***Measure*-36**, ***Measure*-37**, ***Measure*-47**, ***Measure*-49**, ***Measure*-62**.

12.4.6 Attack Simulation Test

In many cases, the tester may be lack of some secret information which was used to generate the test data. When some security protection involves a secret key, and the tester does not have the secret key, then the test under this situation can be regarded as blind test.

Blind test is to check whether the claimed security service is in place, i.e., it is to check the existence of the claimed security service. There is no efficient method for blind test, the most practical way seems to be attack simulation test, also known as penetration test. Here the attack simulation means that, if the claimed security service is provided, then it should be resistant against the relevant attack, and the test is to verify whether the attack simulation could possibly break the claimed security service.

With regard to data confidentiality, an attack simulation on data confidentiality is to try to reveal the content of the test data by whatever methods. Due to the security assumption of encryption algorithms, it is difficult to implement such test, simply because there is no realistic method to conduct the test. Hence, the attack simulation method is not suitable to data confidentiality test.

Given the varieties of the attack simulation tests, some specific such tests are described in a more detail as below.

12.4.6.1 Attack Simulation on Data Integrity Service

With respect to the data integrity service, an attack simulation test is clearly trying to break the data integrity. The test can be performed by tempering the application data in some random manner, and check whether or not the tempered data can be detected by the receiver. For such a test to be possible, there should be a visible evidence to show whether the receiver judges a received message as legitimate or not. Such test should be repeated for a number of times, and any successful breach would result in the data integrity service to fail the test.

12.4.6.2 Attack Simulation on Data Source Authentication

With respect to the data source authentication, which is usually done by identity authentication, the mechanism is to check whether the data comes from a legitimate source, which is usually done by verifying that the identity information associated with the data is legitimate. In order for the authentication to be done, there should be authentication tags or other associated data created by using a secret information (a secret key, or a private key) that cannot possibly be created by any other party without the secret. The process of authentication verification is to check the validity of the authentication tag and the identity.

An attack simulation test on data source authentication is to try to change the identity information of a legitimate message sent to the receiver, or to create a message using a fake identity, which may or may not exist in the system. Such test should be repeated for a number of times, and any successful breach would result in the data source authentication service to fail the test.

12.4.6.3 Attack Simulation on Data Freshness

The data freshness service is for the purpose of thwarting replay attack, where historical data can be identified and will not be accepted. This is particularly important when the data is a control command, which may have confidentiality and integrity protection, as well as data source authentication. However, replay attack simply re-send the data captured earlier to the receiver in a later time. When the replayed data is re-sent to the receiver, if no data freshness service is provided,

the receiver would simply go through the integrity checking and authentication verification processes, and all work properly, even the decryption works properly, hence the data is accepted and processed. For a control command data, the receiver will naturally execute the command after the necessary security processing.

The data freshness service provides a mechanism for the receiver to identify that the data is fresh (never received before) or not. It is not necessary for the data to be received before. The most common techniques for data freshness protection include the use of a counter or timestamp, each method has advantages and disadvantages. Considering the implementation cost, timestamp is often used if the communication entities each has a clock.

The attack simulation on data freshness, is to simulate replay attack. The replay attack is simply to capture some legitimate data from the system, then re-send the data to the receiver in a later time, and see whether the data is accepted by the receiver. This attack is conducted for a number of times, if any such replayed message is accepted by the receiver, then the system can be judged as failing the test.

The above replay attack can be prevented by employing a timestamp or a counter, such data used for freshness verification is called *freshness tag*. The receiver can tell whether the data is valid/fresh by simply checking the validity of the freshness tag.

A more sophisticated replay attack simulation is as follows: It first should identify which part of the captured data is the freshness tag. If it is a counter, then increase the counter value before performing the replay attack. If it is a timestamp, then replace the data with the current timestamp extracted from the machine conducting the attack simulation. Such a replay attack is called *active replay attack*, i.e., the replay attack involves active and sophisticated data modification.

In order to thwart the active replay attack, a common mechanism is to protect the freshness tag. For example, let the freshness tag be part of data to be encrypted, or to have integrity protection, or to be digitally signed. In such a case, any modification on the freshness tag can be found by the receiver. Under such security protection, the active replay attack simulation will unlikely to be able to breach the system.

12.5 Test Methods of Some Specific Security Functionalities

The security measures can be met by some security techniques, including security functions, security mechanisms (e.g. a specific security protocol design), security management, and manpower involvement. From the supervision point of view, security test is a necessary process to confirm whether or not a security measure is met, so there should be corresponding security test methods for the above listed security measures.

However, it is realized that, there may be a number of different test methods for a same security measure, and a same test method may be applicable to a number of security measures. This section studies the test methods of security measures based on some security functions.

12.5.1 Test Methods of Data Encryption

The data encryption functionality can provide a number of security requirements. The preliminary security service provided by an encryption algorithm is to protect data confidentiality, avoiding the content of the protected data to be illegally revealed. As we can see from Sect. 12.6, an encryption algorithm can also provide other security services in certain circumstances, such as to protect the data integrity, provide data source authentication, etc.

Now let's see how to test whether an encryption algorithm is as it is supposed to be.

The test on encryption algorithm may be under the condition of unknown key, hence the test becomes to check whether the target data "looks like" a ciphertext. It is known that a ciphertext of a good cipher is indistinguishable from a random number, hence the test is to check whether the target data has good randomness like random numbers. This is a kind of randomness test. However, the target data is a specific ciphertext, which does not have randomness property. In order for the randomness test to work, there should be a sufficient number of target data which are supposed to be ciphertexts, and the test method is to check whether such data are with good randomness property.

There are many randomness test methods, many classical randomness test methods are adopted by the NIST standard [7]. However, those traditional randomness test methods requires sufficiently large amount of date for test (called data sample), where in IoT environment, to get that amount of data required by the randomness test may not be possible. One may also run the randomness test algorithms on small amount of test data, but the correctness of test result may be too far from being accurate, in the sense of large false positive and large false negative. The problem of randomness test on IoT data with small amount of data sample is a challenging research problem.

If the encryption key is given, then the test is simply to verify the correctness of the ciphertext by performing the decryption algorithm. If the decryption algorithm with the given key yields a plaintext that makes sense, i.e., the decrypted text conforms with the application data in format. In practical test, plaintext-ciphertext pairs are also given.

In order to differentiate the two test cases, we call the test on encryption functionality without the key as *existence test of encryption*, i.e., the test tells whether the encryption functionality exists or not. The test on encryption functionality when the key is given is called *correctness test on encryption*, i.e., the test tells whether the encryption algorithm is a valid one.

12.5.2 Test Methods of Message Authentication Code

The message authentication code (MAC) is a common method to provide data integrity service. The mechanism of MAC algorithm is to map the target data into a larger message space, where the rule of mapping is defined by a key shared by the sender and the receiver of the message. The purpose of MAC is that, any tempering attempt on the mapped message will likely to be detected by the receiver, since the modification is likely to break the rule of the key-controlled mapping.

The most common MAC algorithms are hash function based. For example, FIPS PUB 198-1 defines HMAC as message authentication code. Other kinds of message authentication code algorithms include OMAC. A hash function based MAC algorithm works like this: given the data and the key for the MAC algorithms, the MAC algorithm generates a piece of data called message authentication tag, which is attached to the original data and transmitted to the receiver together with the the original data. If the data being transmitted is tempered, regardless whether the tempering is on the original data, or the tag data, or both, the tempered tag is likely not to match the tempered data, hence can be detected by the receiver.

The test of MAC makes sense only when the MAC algorithm is known, which means that the message authentication tag can be extracted from the test data. If the key is not given, then the test is to check the randomness of the tag data, since the output of a good MAC algorithm is indistinguishable from a random number. The randomness test is like the case for ciphertexts, where the data sample in the IoT scenario is usually small, and effective randomness test algorithm should be studied further.

If the key for the MAC algorithm is given, then the test is simply to check whether the tag data matches the main data by performing the process of generating the tag data.

In order to differentiate these two test cases, the test without knowing the key is called *existence test of MAC*, while the test with the key being given is called *correctness test of MAC*.

12.5.3 Test Methods of Identity Authentication

The concept of identity authentication is the same as that of message source authentication, because the mechanism of the authentication is to verify whether the claimed identity (in case of data source authentication, the identity of the sender or the address of the data source) is valid.

If the authentication is done using a digital signature, then the test process is simply to verify the validity of the signature. Very often the identity authentication service is done using an authentication protocol, where a few rounds of message exchange between the sender and the receiver complying with pre-define rules (what the protocol defines) are involved, in such case, there are different situations to

conduct the test: (1) The tester has all the necessary information to conduct the verification process as what the data receiver does, in this case, the process of the test is simply to repeat the process of what the receiver does in verifying the validity of the data. For example, if a message is associated with a digital signature from the claimed sender, then the test process is simply to verify the validity of the signature. If it is valid, then authentication is passed, otherwise, it means that the authentication fails. If the verification process requires a secret key, and the tester knows such key, then the test process is also as simple as the verification process by the receiver. (2) The verification process requires a secret, but the tester does not know the secret key. In this case, the best way to conduct the test is probably the attack simulation test as described in Sect. 12.4.6.

12.6 An Example of Lightweight Security Implementation and Its Security Test

Typical security requirements include data content protection, data integrity protection, and data source authentication. A security function is meant to provide a method for some security requirements. For example, encryption algorithms provide services for data content protection, message authentication codes provide services for data integrity protection, and digital signature algorithms provide services for data source authentication. However, the relations between security functions and security requirements are not one-to-one. For example, an encryption algorithm, if properly used, can provide data content protection service, data integrity service, and data source authentication service at the same time. It may provide other security services as well, such as data freshness.

12.6.1 A Lightweight Security Protocol

For example, assume that user A and user B share a secret key k. When A wants to transmit a small data m to B, they follow the following steps:

1. A computes: $C = E_k(ID_A, T, m)$, where E is a symmetric key encryption algorithm such as AES [1] that both A and B know in advance, ID_A and ID_B are the identities of A and B respectively, T is a timestamp.
2. A sends to B the following: ID_A, ID_B, T, C.
3. B does the following:
   (i) Finds the decryption key k with the information ID_A, decrypts C to get ID_A, T and m.
   (ii) Checks whether the decrypted ID_A and T are the same as those in plaintext part. If not, then stops further processing.

(iii) Checks whether T is within an allowable time slot. If not, then stops processing.
(iv) Accepts m and processes it as so needed.

12.6.2 The Security Services That the Protocol Can Provide

It is easy to verify that, the above simple security protocol provides data confidentiality, because the message m is encrypted. Intercepting the transmitted messages do not enable the recovery of m without knowing the encryption key k (which is also the decryption key in the case of symmetric key cipher).

It is also noted that, if one tries to modify the ciphertext C into C', then when C' is decrypted by B, the part which is supposed to be ID_A and T is very likely to be something else, i.e., the decrypted ID'_A and T' will not match those in the plaintext, and the chances for exception is negligible. This means that the above simple protocol provides data integrity service. It is noted that the data integrity service is achieved by using an encryption algorithm plus the redundancy of the plaintext data.

The use of timestamp T is to provide data freshness service. When T is not in an allowable time slot, then the message cannot be accepted. The data freshness service provides the ability to thwart replay attacks.

In addition to the above described security services, the above simple protocol also provides data source authentication. The identities of A and B indicate that the message comes from A, and correct format of ID_A after decryption using the secret key k fulfills the authentication, because k is shared only by A and B.

The above example means that, by doing one encryption, it is possible to provide data content protection (i.e., data confidentiality), data integrity, data source authentication, and data freshness as well. On the other hand, there may be more than one security function to achieve the same security requirement. For example, encryption algorithms, message authentication codes, and digital signature algorithms, can all be used to provide data integrity service.

12.6.3 Security Test on the Lightweight Security Protocol

With respect to the above lightweight security protocol, how to test the functionalities that it provides is an interesting but not trivial question. Basically, we mainly consider how to test the confidentiality, integrity, authentication, and data freshness services.

Traditionally, the security test on the confidentiality is to check whether the data is in encrypted form. The confidentiality service only applies to the application data, not the affiliated data such as identity information. In the above protocol, it is to test whether C is a ciphertext. This can be done using the ciphertext-only test or the known-key test.

With respect to the integrity test, traditionally it is to check the existence of a message tag, because message integrity service is usually provided by a message authentication code algorithm. However, this test method is inappropriate to IoT data. It is recommended to test the integrity functionality, not to connect the functionality with any specific method of implementation. Under this principle, the penetration test seems to be more appropriate. However, if the encryption/decryption key is given, by checking the correct decryption of the message, and the correct format of the message, and the message integrity protection mechanism provided by the security solution, the integrity test should naturally be passed.

With respect to the security test on the message source authentication and the test on the data freshness, if the encryption/decryption key is unknown, the most appropriate method is still the penetration test, i.e., to simulate attacks on the test data sample. If the encryption/decryption key is given, then similar to the case of confidentiality test, the correct decryption of the message, the correct format and content of the message, and the mechanism of data source authentication and that of data freshness services provided by the security solution, form the evidence of the test result. The mechanism of data source authentication is the fact that only the correct sender of the message has the correct encryption key, and hence can possibly create the correct ciphertext. The mechanism of data freshness is that, the timestamp T is protected by an encryption algorithm so that it cannot be changed without the key. Any tempering will almost definitely be detectable.

In summary, the lightweight security protocol gives an example to show that, the security test on many security services or functionalities should focus on the security services and functionalities, and not to assume any specific implementation method.

12.7 Summary

Although the fundamental information security services are data confidentiality, data integrity, and availability (known as CIA triad), the actual security services in practical IoT application systems can be much more and specific. This is because the practical application systems have to deal with the details as how to protect the data, the users and creators of the data, the process of data transmission, as well as the systems processing the data.

In designing security functionalities for an IoT system, it is not surprising to miss some important security measures. The purpose of this chapter is meant to provide a resource of reference for IoT security designers and security standard makers.

This chapter presents some 60 security measures for IoT systems, mainly focusing on the IoT perception layer. The security measures are made from different angles, so one security measure may cover another with a higher (or a lower) level of requirement. It is not practically possible to implement all the security measures in an IoT application, a suitable subset of the security measures is sufficient.

It should be noted that, although this chapter tries to make the security measures comprehensive, it is however by no means complete. More work is needed to enrich the list of security measures, and completion of security measures is a continuous work. It is expected that, with a specific IoT application, more specific security measures can be found, or some of the listed security measures can be made more specific in detail, while it is unavoidable that some of the measures in the list become inappropriate.

References

1. *Advanced Encryption Standard*, FIPS PUB 197, National Institute of Standards and Technology (NIST), 2001
2. Y. Deswarte, L. Blain, J.C. Fabre, Intrusion tolerance in distributed computing systems, in *Proc. of the IEEE Symposium on Research in Security and Privacy, Oakland*, 20–22 May 1991, pp. 110–122
3. D. Eastlake, T. Hansen, US Secure Hash Algorithms (SHA and HMAC-SHA), RFC 4634, July 2006
4. M. Grabovica, S. Popic, D. Pezer, V. Knezevic, Provided security measures of enabling technologies in Internet of Things (IoT): a survey, in *Proceedings of the Zooming Innovation in Consumer Electronics International Conference (ZINC), Novi Sad*, 1–2 June 2016, pp. 28–31
5. J. King, A.I. Awad, A distributed security mechanism for resource-constrained IoT devices. Informatica **40**, 133–143 (2016)
6. S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, Y. Halgand, A survey of approaches combining safety and security for industrial control systems. Reliab. Eng. Syst. Saf. **139**, 156–178 (2015)
7. E. Lawrence et al., A statistical test suite for random and pseudorandom number generators for cryptographic applications, special publication (NIST SP) - 800-22 Rev 1a. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf
8. R.M. Needham, Denial of service: an example. Commun. ACM **37**(11), 42–46 (1994)
9. R.L. Rivest, The MD5 message-digest algorithm. RfC 1321, April 1992
10. C.E. Shannon, Communication theory of secrecy systems. Bell Syst. Tech. J. **28**, 656–715 (1949)
11. D. Zhang, L.T. Yang, M. Chen, S. Zhao, M. Guo, Y. Zhang, Real-time locating systems using active RFID for Internet of Things. IEEE Syst. J. **10**(3), 1226–1235 (2016)
12. W. Zhao, Intrusion tolerance techniques, in *Encyclopedia of Information Science and Technology*, 4th edn. (IGI Global, Hershey, 2018), pp. 4927–4936