CHAPTER 23

Internet of Things (IoT) Security

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- Explain the scope of the Internet of Things.
- List and discuss the five principal components of IoT-enabled things.
- Understand the relationship between cloud computing and IoT.
- Define the patching vulnerability.
- Explain the IoT Security Framework.
- Understand the MiniSec security feature for wireless sensor networks.

This chapter begins with an overview of the concepts of the IoT, followed by a discussion of IoT security.

23.1 THE INTERNET OF THINGS

The Internet of Things is the latest development in the long and continuing revolution of computing and communications. Its size, ubiquity, and influence on everyday lives, business, and government dwarf any technical advance that has gone before. This section provides a brief overview of the Internet of Things.

Things on the Internet of Things

The **Internet of Things (IoT)** is a term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors. A dominant theme is the embedding of short-range mobile transceivers into a wide array of gadgets and everyday items, enabling new forms of communication between people and things, and between things themselves. The Internet now supports the interconnection of billions of industrial and personal objects, usually through cloud systems. The objects deliver sensor information, act on their environment, and in some cases modify themselves, to create overall management of a larger system, like a factory or city.

The IoT is primarily driven by deeply embedded devices. These devices are low-bandwidth, low-repetition data capture and low-bandwidth data-usage appliances that communicate with each other and provide data via user interfaces. Embedded appliances, such as high-resolution video security cameras, video VoIP phones, and a handful of others, require high-bandwidth streaming capabilities. Yet countless products simply require packets of data to be intermittently delivered.

Evolution

With reference to the end systems supported, the Internet has gone through roughly four generations of deployment culminating in the IoT:

1. **Information technology (IT):** PCs, servers, routers, firewalls, and so on, bought as IT devices by enterprise IT people, primarily using wired connectivity.
2. **Operational technology (OT):** Machines/appliances with embedded IT built by non-IT companies, such as medical machinery, SCADA (supervisory control and data acquisition), process control, and kiosks, bought as appliances by enterprise OT people and primarily using wired connectivity.
3. **Personal technology:** Smartphones, tablets, and eBook readers bought as IT devices by consumers (employees) exclusively using wireless connectivity and often multiple forms of wireless connectivity.
4. **Sensor/actuator technology:** Single-purpose devices bought by consumers, IT, and OT people exclusively using wireless connectivity, generally of a single form, as part of larger systems.

It is the fourth generation that is usually thought of as the IoT, and which is marked by the use of billions of embedded devices.

Components of IoT-Enabled Things

The key components of an IoT-enabled device are the following (Figure 23.1):

- **Sensor:** A sensor measures some parameter of a physical, chemical, or biological entity and delivers an electronic signal proportional to the observed characteristic, either in the form of an analog voltage level or a digital signal.

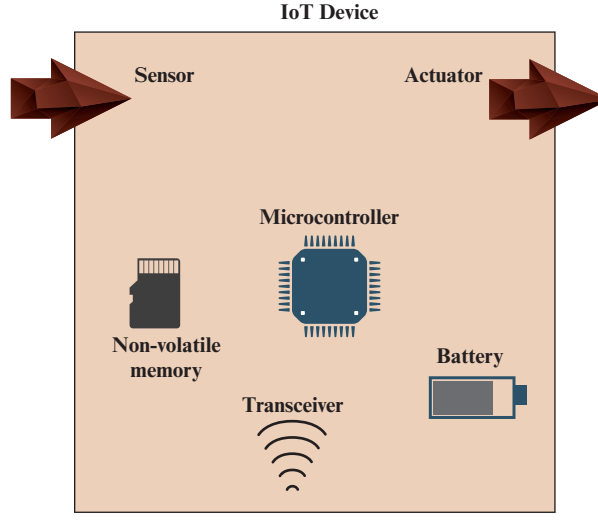


Figure 23.1 IoT Components

In both cases, the sensor output is typically input to a microcontroller or other management element.

- **Actuator:** An actuator receives an electronic signal from a controller and responds by interacting with its environment to produce an effect on some parameter of a physical, chemical, or biological entity.
- **Microcontroller:** The "smart" in a smart device is provided by a deeply embedded microcontroller.
- **Transceiver:** A transceiver contains the electronics needed to transmit and receive data. Most IoT devices contain a wireless transceiver, capable of communication using Wi-Fi, ZigBee, or some other wireless scheme.
- **Power supply:** Typically, this is a battery.

IoT devices also often contain a Radio-Frequency Identification (RFID) component. RFID technology, which uses radio waves to identify items, is increasingly becoming an enabling technology for IoT. The main elements of an RFID system are tags and readers. RFID tags are small programmable devices used for object, animal, and human tracking. They come in a variety of shapes, sizes, functionalities, and costs. RFID readers acquire and sometimes rewrite information stored on RFID tags that come within operating range (a few inches up to several feet). Readers are usually connected to a computer system that records and formats the acquired information for further uses.

IoT and Cloud Context

To better understand the function of an IoT, it is useful to view it in the context of a complete enterprise network that includes third-party networking and cloud computing elements. Figure 23.2 provides an overview illustration.

Edge At the **edge** of a typical enterprise network is a network of IoT-enabled devices, consisting of sensors and perhaps actuators. These devices may communicate with one another. For example, a cluster of sensors may all transmit their data to one sensor that aggregates the data to be collected by a higher-level entity. At this level also there may also be a number of gateways. A gateway interconnects the IoT-enabled devices with the higher-level communication networks. It performs the necessary translation between the protocols used in the communication networks and those used by devices. It may also perform a basic data aggregation function.

Fog In many IoT deployments, massive amounts of data may be generated by a distributed network of sensors. For example, offshore oil fields and refineries can generate a terabyte of data per day. An airplane can create multiple terabytes of data per hour. Rather than store all of that data permanently (or at least for a long period) in central storage accessible to IoT applications, it is often desirable to do as much data processing close to the sensors as possible. Thus, the purpose of what is sometimes referred to as the fog computing level is to convert network data flows into information that is suitable for storage and higher level processing. Processing elements at this level may deal with high volumes of data and perform

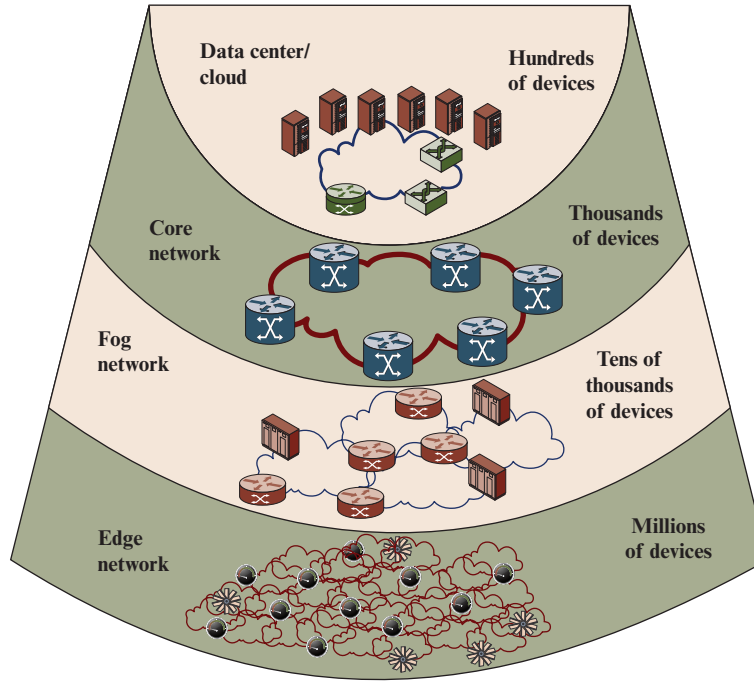


Figure 23.2 The IoT/Cloud Context

data transformation operations, resulting in the storage of much lower volumes of data. The following are examples of **fog** computing operations:

- **Evaluation:** Evaluating data for criteria as to whether it should be processed at a higher level.
- **Formatting:** Reformatting data for consistent higher-level processing.
- **Expanding/decoding:** Handling cryptic data with additional context (such as the origin).
- **Distillation/reduction:** Reducing and/or summarizing data to minimize the impact of data and traffic on the network and higher-level processing systems.
- **Assessment:** Determining whether data represent a threshold or alert; this could include redirecting data to additional destinations.

Generally, fog computing devices are deployed physically near the edge of the IoT network; that is, near the sensors and other data-generating devices. Thus, some of the basic processing of large volumes of generated data is offloaded and outsourced from IoT application software located at the center.

Fog computing and fog services are expected to be a distinguishing characteristic of the IoT. Fog computing represents an opposite trend in modern networking from cloud computing. With cloud computing, massive, centralized storage and processing resources are made available to distributed customers over cloud networking

facilities to a relatively small number of users. With fog computing, massive numbers of individual smart objects are interconnected with fog networking facilities that provide processing and storage resources close to the edge devices in an IoT. Fog computing addresses the challenges raised by the activity of thousands or millions of smart devices, including security, privacy, network capacity constraints, and latency requirements. The term *fog computing* is inspired by the fact that fog tends to hover low to the ground whereas clouds are high in the sky.

Core The **core** network, also referred to as a **backbone network**, connects geographically dispersed fog networks as well as provides access to other networks that are not part of the enterprise network. Typically, the core network will use very high performance routers, high-capacity transmission lines, and multiple interconnected routers for increased redundancy and capacity. The core network may also connect to high-performance, high-capacity servers, such as large database servers and private cloud facilities. Some of the core routers may be purely internal, providing redundancy and additional capacity without serving as edge routers.

Cloud The **cloud** network provides storage and processing capabilities for the massive amounts of aggregated data that originate in IoT-enabled devices at the edge. Cloud servers also host the applications that interact with and manage the IoT devices and that analyze the IoT-generated data.

Table 23.1 compares cloud and fog computing.

Table 23.1 Comparison of Cloud and Fog Features

	Cloud	Fog
Location of processing/storage resources	Center	Edge
Latency	High	Low
Access	Fixed or wireless	Mainly wireless
Support for mobility	Not applicable	Yes
Control	Centralized/hierarchical (full control)	Distributed/hierarchical (partial control)
Service access	Through core	At the edge/on handheld device
Availability	99.99%	Highly volatile/highly redundant
Number of users/devices	Tens/hundreds of millions	Tens of billions
Main content generator	Human	Devices/sensors
Content generation	Central location	Anywhere
Content consumption	End device	Anywhere
Software virtual infrastructure	Central enterprise servers	User devices

23.2 IOT SECURITY CONCEPTS AND OBJECTIVES

IoT is perhaps the most complex and undeveloped area of network security. To see this, consider Figure 23.3, which shows the main elements of interest for IoT security. At the center of the network are the application platforms, data storage servers, and network and security management systems. These central systems gather data from sensors, send control signals to actuators, and are responsible for managing the IoT devices and their communication networks. At the edge of the network are IoT-enable devices, some of which are quite simple, constrained devices and some of which are more intelligent, unconstrained devices. In addition, gateways may perform protocol conversion and other networking service on behalf of IoT devices.

Figure 23.3 illustrates a number of typical scenarios for interconnection and the inclusion of security features. The shading in Figure 23.3 indicates the systems that support at least some of these functions. Typically, gateways will implement secure functions, such as TLS and IPsec. Unconstrained devices may or may not implement some security capability. Constrained devices generally have limited or no security features. As suggested in the figure, gateway devices can provide secure communication between the gateway and the devices at the center, such as application platforms and management platforms. However, any constrained or unconstrained devices attached to the gateway are outside the zone of security established between the gateway and the central systems. As shown, unconstrained devices can communicate directly with the center and support security functions. However, constrained devices that are not connected to gateways have no secure communications with central devices.

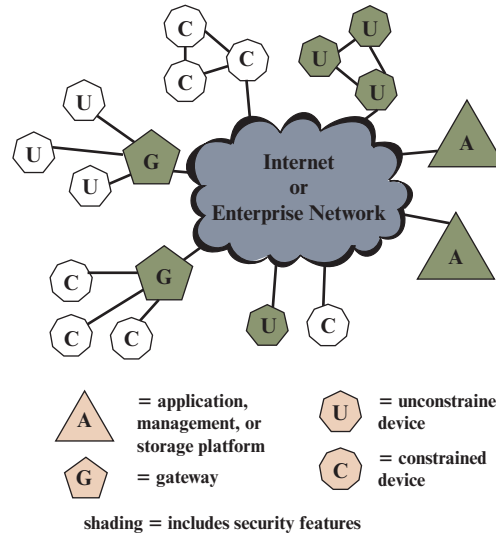


Figure 23.3 IoT Security: Elements of Interest

Unique Characteristics of the IoT Ecosystem

The European Union Agency For Network And Information Security (ENISA) *Baseline Security Recommendations for IoT* [ENIS17] lists the following issues that hinder the development of secure IoT ecosystems:

- **Very large attack surfaces:** This topic is explored later in this section. In essence, there are a wide variety of points of vulnerability within an IoT ecosystem and a large variety of data that may be compromised.
- **Limited device resources:** IoT devices are typically constrained devices, with limited memory, processing power, and power supply.

  This makes it difficult to employ advanced security controls.
- **Complex ecosystem:** The IoT involves not only a large number of devices, but the interconnections, communications, and dependencies among them and with cloud elements. This makes the task of assessing security risk extremely complex.
- **Fragmentation of standards and regulations:** Comparatively little work has been done on security standards for IoT, as well as limited best practices documentation. Thus, there is a lack of comprehensive guidance for security managers and implementers.
- **Widespread deployment:** There is an ongoing rapid deployment of IoT arrangements in commercial environments and, more importantly, critical infrastructure environments. These deployments are attractive targets for security attacks and the rapid deployment is often without comprehensive risk assessment and security planning.
- **Security integration:** IoT devices use a wide variety of communications protocols, and when implemented, authentication schemes. In addition, there may be contractor viewpoints and requirements from all involved stakeholders. Integrating security into an interoperable scheme is thus extraordinarily challenging.
- **Safety aspects:** Because many IoT devices act on their physical environment, security threats can become safety threats, raising the bar for the effectiveness of security solutions.
- **Low cost:** IoT devices are manufactured, purchased, and deployed in millions. This provides great incentive for all parties to minimize the cost of these devices. Manufacturers might be inclined to limit security features to maintain a low cost, and customers might be inclined to accept these limitations.
- **Lack of expertise:** IoT is still a relatively new and rapidly evolving technology. There are a limited number of people with suitable cybersecurity training and experience.
- **Security updates:** In an often-quoted 2014 article, security expert Bruce Schneier stated that we are at a crisis point with regard to the security of embedded systems, including IoT devices [SCHN14]. The embedded devices are riddled with vulnerabilities and there is no good way to patch them. The chip manufacturers have strong incentives to produce their product with its

firmware and software as quickly and cheaply as possible. The device manufacturers choose a chip based on price and features and do very little if anything to the chip software and firmware. Their focus is the functionality of the device itself. The end user may have no means of patching the system or, if so, little information about when and how to patch. The result is that the hundreds of millions of Internet-connected devices in the IoT are vulnerable to attack. This is certainly a problem with sensors, allowing attackers to insert false data into the network. It is potentially a graver threat with actuators, where the attacker can affect the operation of machinery and other devices.

- **Insecure programming:** Effective cybersecurity practice requires the integration of security planning and design throughout the software development lifecycle. But again, with cost pressure, developers of IoT products have an incentive to place more emphasis on functionality and usability than on security.
- **Unclear liabilities:** A major IoT deployment involves a large and complex supply chain and complex interaction among numerous components. Because it is difficult under these circumstances to clearly assign liabilities, ambiguities and conflicts may arise in the event of a security incident.

IoT Security Objectives

NISTIR 8200 (*Interagency Report on Status of International Cybersecurity Standardization for the Internet of Things*) lists the following security objectives for IoT:

- **Restricting logical access to the IoT network.** This may include: using unidirectional gateways, using firewalls to prevent network traffic from passing directly between the corporate and IoT networks, and having separate authentication mechanisms and credentials for users of the corporate and IoT networks. An IoT system should also use a network topology that has multiple layers, with the most critical communications occurring in the most secure and reliable layer.
- **Restricting physical access to IoT network and components.** A combination of physical access controls should be used, such as locks, card readers, and/or guards.
- **Protecting individual IoT components from exploitation.** This includes deploying security patches in as expeditious a manner as possible, after testing them under field conditions; disabling all unused ports and services and assuring that they remain disabled; restricting IoT user privileges to only those that are required for each person's role; tracking and monitoring audit trails; and using security controls such as antivirus software and file integrity checking software where technically feasible.
- **Preventing unauthorized modification of data.** This includes data that are in transit (at least across the network boundaries) and at rest.
- **Detecting security events and incidents.** The object is to detect security events early enough to break the attack chain before attackers attain their objectives. This includes the capability to detect failed IoT components, unavailable services, and exhausted resources that are important to provide proper and safe functioning of an IoT system.

- **Maintaining functionality during adverse conditions.** This involves designing IoT systems so that each critical component has a redundant counterpart. Additionally, if a component fails, it should fail in a manner that does not generate unnecessary traffic on IoT or other networks, or does not cause another problem elsewhere. IoT systems should also allow for graceful degradation such as moving from normal operation with full automation to emergency operation with operators more involved and less automation to manual operation with no automation.
- **Restoring the system after an incident.** Incidents are inevitable and an incident response plan is essential. A major characteristic of a good security program is how quickly the IoT system can be recovered after an incident has occurred.

Tamper Resistance and Detection

An IoT ecosystem involves a large number of devices deployed in the edge network and in the fog network. Typically these involve numerous manufacturers and multiple supply chains and often deployment in areas where physical security is difficult. Two essential security measures in such an environment are tamper resistance and tamper detection. We define the following terms:

- **Tampering:** An unauthorized modification that alters the intended functioning of a system or device in a way that degrades the security it provides.
- **Tamper resistant:** A characteristic of a system component that provides passive protection against an attack.
- **Tamper detection:** Techniques to ensure that the overall system is made aware of unwanted physical access.

Tamper Resistance The common approach to tamper resistance is to use specialized physical construction materials to make tampering with a fog node difficult. Examples include hardened steel enclosures, locks, and security screws. Tightly packing components and circuit boards within an enclosure increases the difficulty of using fiber optics to probe inside the node without opening the enclosure.

A second category of tamper resistance is the deterrence of tampering by ensuring that tampering leaves visible evidence behind. Examples include special seals and tapes that make it obvious when there has been physical tampering.

Tamper Detection Mechanisms for tamper detection include the following:

- **Switches:** A variety of switches, such as mercury switches, magnetic switches, and pressure contacts can detect the opening of a device, the breach of a physical security boundary, or the movement of a device.
- **Sensors:** Temperature and radiation sensors can detect environmental changes. Voltage and power sensors can detect electrical attacks.
- **Circuitry:** It is possible to wrap components with flexible circuitry, resistance wire, or fiber optics so as to detect a puncture or break.

Gateway Security

ITU-T Recommendation Y.2066 (Common Requirements of the Internet of Things, June 2014) includes a list of security requirements for the IoT. This list is a useful baseline for understand the scope of security implementation needed for an IoT deployment. The requirements are defined as being the functional requirements during capturing, storing, transferring, aggregating and processing the data of things, as well as to the provision of services which involve things. These requirements are related to all the IoT actors. The requirements are:

- **Communication security:** Secure, trusted, and privacy-protected communication capability is required, so that unauthorized access to the content of data can be prohibited, integrity of data can be guaranteed, and privacy-related content of data can be protected during data transmission or transfer in IoT.
- **Data management security:** Secure, trusted, and privacy-protected data management capability is required, so that unauthorized access to the content of data can be prohibited, integrity of data can be guaranteed, and privacy-related content of data can be protected when storing or processing data in IoT.
- **Service provision security:** Secure, trusted, and privacy-protected service provision capability is required, so that unauthorized access to service and fraudulent service provision can be prohibited and privacy information related to IoT users can be protected.
- **Integration of security policies and techniques:** The ability to integrate different security policies and techniques is required, so as to ensure a consistent security control over the variety of devices and user networks in IoT.
- **Mutual authentication and authorization:** Before a device (or an IoT user) can access the IoT, mutual authentication and authorization between the device (or the IoT user) and IoT is required to be performed according to predefined security policies.
- **Security audit:** Security audit is required to be supported in IoT. Any data access or attempt to access IoT applications are required to be fully transparent, traceable, and reproducible according to appropriate regulation and laws. In particular, IoT is required to support security audit for data transmission, storage, processing, and application access.

A key element in providing security in an IoT deployment is the gateway. Y.2067 (*Common Requirements and Capabilities of a Gateway for Internet of Things Applications*, June 2014) details specific security functions that the gateway should implement, some of which are illustrated in Figure 23.4. These consist of the following:

- Support identification of each access to the connected devices.
- Support authentication with devices. Based on application requirements and device capabilities, it is required to support mutual or one-way authentication with devices. With one-way authentication, either the device authenticates itself to the gateway or the gateway authenticates itself to the device, but not both.
- Support mutual authentication with applications.

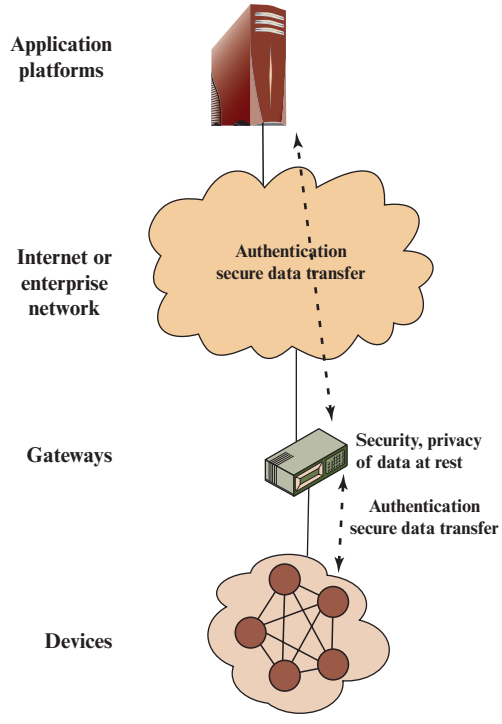


Figure 23.4 IoT Gateway Security Functions

- Support the security of the data that are stored in devices and the gateway, or transferred between the gateway and devices, or transferred between the gateway and applications. Support the security of these data based on security levels.
- Support mechanisms to protect privacy for devices and the gateway.
- Support self-diagnosis and self-repair as well as remote maintenance.
- Support firmware and software update.
- Support auto configuration or configuration by applications. The gateway is required to support multiple configuration modes, e.g., remote and local configuration, automatic and manual configuration, and dynamic configuration based on policies.

Some of these requirements may be difficult to achieve when they involve providing security services for constrained devices. For example, the gateway should support security of data stored in devices. Without encryption capability at the constrained device, this may be impractical to achieve.

Note that the Y.2067 requirements make a number of references to privacy requirements. Privacy is an area of growing concern with the widespread deployment of IoT-enabled things in homes, retail outlets, and vehicles and humans. As more things are interconnected, governments and private enterprises will collect massive amounts of data about individuals, including medical information, location and movement information, and application usage.

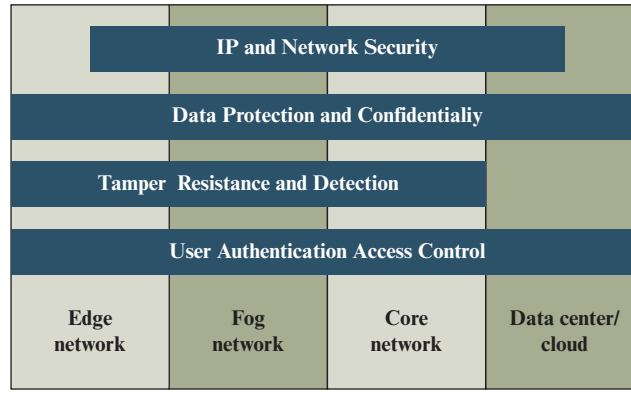


Figure 23.5 IoT Security Environment

The IoT Security Environment

Figure 23.5 models the scope of key security capabilities across the four levels of the IoT ecosystem:

- **User authentication and access control:** These functions span then entire IoT ecosystem. A common approach to access control is role-based access control (RBAC). RBAC systems assign access rights to roles instead of individual users. In turn, users are assigned to different roles, either statically or dynamically, according to their responsibilities. RBAC enjoys widespread commercial use in cloud and enterprise systems and is a well-understood tool that can be used to manage access to IoT devices and the data they generate.
- **Tamper resistance and detection:** This function is particularly important at the device and fog network levels but also extends to the core network level. All of these levels may involve components that are physically outside the area of the enterprise that is protected by physical security measures.
- **Data protection and confidentiality:** These functions extend to all levels of the architecture.
- **Internet protocol and network security:** Protection of data in motion from eavesdropping and snooping is essential between all levels.

23.3 AN OPEN-SOURCE IOT SECURITY MODULE

This section provides an overview of MiniSec, an open-source security module that is part of the TinyOS operating system. TinyOS is designed for small embedded systems with tight requirements on memory, processing time, real-time response, and power consumption. TinyOS takes the process of streamlining quite far, resulting in a very minimal OS for embedded systems, with a typical configuration requiring 48 KB of code and 10 KB of RAM [LEVI12]. The main application of TinyOS is wireless sensor networks and it has become the de facto OS for such networks. With sensor networks, the primary security concerns relate to wireless

communications. MiniSec is designed to be a link-level module that offers a high level of security, while simultaneously keeping energy consumption low and using very little memory [LUK07]. MiniSec provides confidentiality, authentication, and replay protection.

MiniSec has two operating modes, one tailored for single-source communication, and another tailored for multi-source broadcast communication. The latter does not require per-sender state for replay protection and thus scales to large networks.

MiniSec is designed to meet the following requirements:

- **Data authentication:** Enables a legitimate node to verify whether a message originated from another legitimate node (i.e., a node with which it shares a secret key) and was unchanged during transmission.
- **Confidentiality:** A basic requirement for any secure communications system.
- **Replay protection:** Prevents an attacker from successfully recording a packet and replaying it at a later time.
- **Freshness:** Because sensor nodes often stream time-varying measurements, providing guarantee of message freshness is an important property. There are two types of freshness: strong freshness and weak freshness. MiniSec provides a mechanism to guarantee weak freshness, where a receiver can determine a partial ordering over received messages without a local reference time point.
- **Low energy overhead:** This is achieved by minimizing communication overhead and by the use of only symmetric.
- **Resilient to lost messages:** The relatively high occurrence of dropped packets in wireless sensor networks requires a design that can tolerate high message loss rates.

Cryptographic Algorithms

Two cryptographic algorithms used by MiniSec are worth noting. The first of these is the encryption algorithm Skipjack. Skipjack was developed in the 1990s by the U.S. National Security Agency (NSA). It is one of the simplest and fastest block cipher algorithms, which is critical to embedded systems. A study of eight possible candidate algorithms for wireless security networks [LAW06] concluded that Skipjack was the best algorithm in terms of code memory, data memory, encryption/decryption efficiency, and key setup efficiency.

Skipjack makes use of an 80-bit key. It was intended by NSA to provide a secure system once it became clear that DES, with only a 56-bit key, was vulnerable. Contemporary algorithms, such as AES, employ a key length of at least 128 bits, and 80 bits is generally considered inadequate. However, for the limited application of wireless sensor networks and other IoT devices, which provide large volumes of short data blocks over a slow data link, Skipjack may suffice. With its efficient computation and low memory footprint, Skipjack is an attractive choice for IoT devices. However, going forward, it is advisable for any IoT security module to use one of the recently developed lightweight cryptographic algorithms, such as the Scalable Encryption Algorithm (SEA) described in Chapter 14.

The block cipher mode of operation chosen for MiniSec is the Offset Codebook Mode (OCB), described later in this section.

MiniSec employs per-device keys; that is, each key is unique to a particular pair of devices, to prevent replay attacks.

Operating Modes

MiniSec has two operating modes: unicast (MiniSec-U) and broadcast (MiniSec-B). Both schemes use OCB with a counter, known as a nonce, that is input along with the plaintext into the encryption algorithm. The least significant bits of the counter are also sent as plaintext to enable synchronization. For both modes, data are transmitted in packets. Each packet includes the encrypted data block, the OCB authentication tag, and the MiniSec counter.

MiniSec-U employs synchronized counters, which require the receiver to keep a local counter for each sender. The strictly monotonically increasing counter guarantees semantic confidentiality.[1] Even if the sender A repeatedly sends the same message, each ciphertext is different since a different counter value is used. Also, once a receiver observes a counter value, it rejects packets with an equal or smaller counter value. Therefore, an attacker cannot replay any packet that the receiver has previously received. If a number of packets are dropped, the sender and receiver engage in a resynchronization protocol.

MiniSec-U cannot be directly used to secure broadcast communication. First, it would be too expensive to run the counter resynchronization protocol among many receivers. Also, if a node were to simultaneously receive packets from a large group of sending nodes, it would need to maintain a counter for each sender, resulting in high memory overhead. Instead, it uses two mechanisms, a timing-based approach and a bloom-filter approach, that defend against replay attacks. First, the time is divided into t-length epochs E1, E2,. . . . Using the current epoch or the previous epoch as nonce for OCB encryption, the replay of messages from older epochs is avoided. The timing approach is augmented with a bloom-filter approach in order to prevent replay attacks inside the current epoch. MiniSec-B uses as nonce element in OCB encryption and bloom-filter key the string $nodeID.E_i.C_{ab}$, where $nodeID$ is the sender node identifier, E_i is the current epoch, and C_{ab} is a shared counter. Every time that a node receives a message, it checks if it belongs to its bloom filter. If the message is not replayed, it is stored in the bloom filter. Else, the node drops it.

For further details on the two operating modes, see [TOBA07].

Offset Codebook Mode

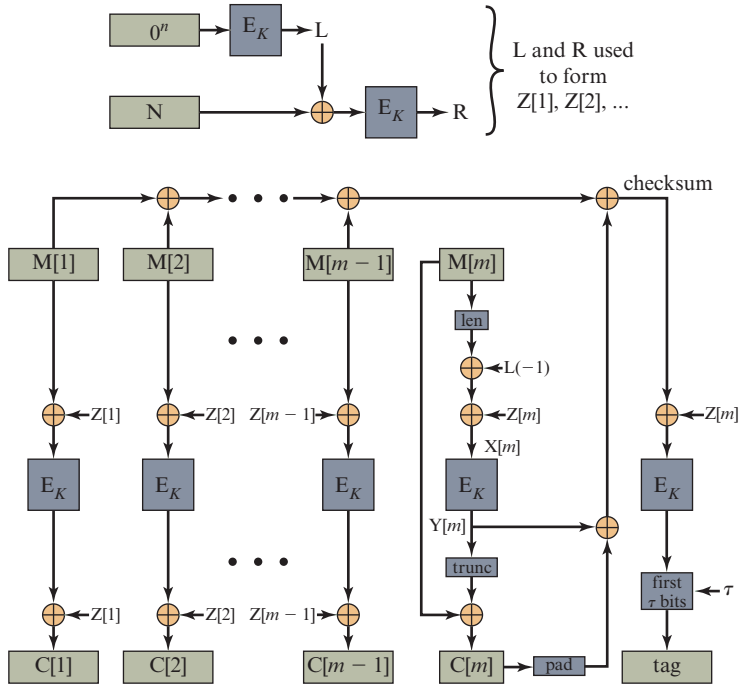
As mentioned in Chapter 7, a mode of operation must be specified when a plaintext source consists of multiple blocks of data to be encrypted with the same encryption key. OCB is an NIST proposed block cipher mode of operation [ROGA01], and is a proposed Internet Standard defined in RFC 7253 (*The OCB Authenticated-Encryption Algorithm*, May 2014). OCB is also approved as an authenticated encryption technique in the IEEE 802.11 wireless LAN standard. And, OCB is included in MiniSec, an open-source IoT security module.

[1]Semantic confidentiality means that if the same plain-text is encrypted twice, the two resulting ciphertexts are different.

A key objective for OCB is efficiency. This is achieved by minimizing the number of encryptions required per message and by allowing for parallel operation on the blocks of a message. OCB mode is provably secure assuming the underlying block cipher is secure. OCB mode is a one-pass mode of operation making it highly efficient. Only one block cipher call is necessary for each plaintext block, with an additional two calls needed to complete the whole encryption process. OCB is especially well suited for the stringent energy constraints of sensor nodes.

Figure 23.6 shows the overall structure for OCB encryption and authentication. Typically, AES is used as the encryption algorithm. The message M to be encrypted and authenticated is divided into n-bit blocks, with the exception of the last block, which may be less than n bits. Typically, $n = 128$. Only a single pass through the message is required to generate both the ciphertext and the authentication code. The total number of blocks is $m = \lceil \text{len}(M)/n \rceil$.

Note that the encryption structure for OCB is similar to that of electronic codebook (ECB) mode. Each block is encrypted independently of the other blocks, so that it is possible to perform all m encryptions simultaneously. As was mentioned in Chapter 7, with ECB, if the same b-bit block of plaintext appears more than once



n = block length in bits
N = nonce
len(M[m]) = length of M[m] represented as an n-bit integer
trunc(Y[m]) = deletes least significant bits so that result is same length as M[m]
pad = pad with least significant 0 bits to length n
τ = length of authentication tag

Figure 23.6 OCB Encryption and Authentication

in the message, it always produces the same ciphertext. Because of this, for lengthy messages, the ECB mode may not be secure. OCB eliminates this property by using an offset Z[i] for each block M[i], such that each Z[i] is unique; the offset is XORed with the plaintext and XORed again with the encrypted output. Thus, with encryption key K we have

$$\mathrm{C}[i] = \mathrm{E}_K(\mathrm{M}[i] \oplus \mathrm{Z}[i]) \oplus \mathrm{Z}[i]$$

where $E_K(X)$ is the encryption of plaintext X using key K, and $\oplus$ is the exclusive-OR operation. Because of the use of the offset, two blocks in the same message that are identical will produce two different ciphertexts.

The upper part of Figure 23.6 indicates how the $Z[i]$ are generated. An arbitrary n-bit value N called the nonce is chosen; the only requirement is that if multiple messages are encrypted with the same key, a different nonce must be used each time such that each nonce is only used once. Each different value of N will produce a different set of $Z[i]$. Thus, if two different messages have identical blocks in the same position in the message, they will produce different ciphertexts because the $Z[i]$ will be different.

The calculation of the $Z[i]$ is somewhat complex and is summarized in the following equations:

$$\begin{aligned}
\mathrm{L}(0) &= \mathrm{L} = \mathrm{E}_\mathrm{K}(0^n) && \text{where } 0^n \text{ is consists of n zero bits.}\\
\mathrm{R} &= \mathrm{E}_K(\mathrm{N} \oplus \mathrm{L}) &&\\
\mathrm{L}(i) &= 2 \cdot \mathrm{L}(i-1) && 1 \le i \le m\\
\mathrm{Z}[1] &= \mathrm{L} \oplus \mathrm{R} &&\\
\mathrm{Z}[i] &= \mathrm{Z}(i-1) \oplus \mathrm{L}(\mathrm{ntz}(i)) && 1 \le \mathrm{i} \le m
\end{aligned}$$

The operator $\cdot$ refers to multiplication over the finite field $\mathrm{GF}(2^n)$. The operator $\mathrm{ntz}(i)$ denotes the number of trailing (least significant) zeros in i. The resulting $\mathrm{Z}[i]$ values are a maximal Hamming distance apart [WALK05].

Thus, the values $\mathrm{Z}[i]$ are a function of both the nonce and the encryption key. The nonce does not need to be kept secret and is communicated to the recipient in a manner outside the scope of the specification.

Because the length of M may not be an integer multiple of n, the final block is treated differently, as shown in Figure 23.6. The length of M[m], represented as an n-bit integer, is used to calculate $\mathrm{X}[m] = \mathrm{len}(\mathrm{M}[m]) \oplus \mathrm{L}(-1) \oplus \mathrm{Z}[m]$. $\mathrm{L}(-1)$ is defined as L/2 over the finite field or, equivalently, $\mathrm{L} \cdot 2^{-1}$. Next, $\mathrm{Y}[m] = \mathrm{E}_K(\mathrm{X}[m])$. Then, Y[$m$] is truncated to len(M[$m$]) bits (by deleting the necessary number of least significant bits) and XORed with M[m]. Thus, the final ciphertext C is the same length as the original plaintext M.

A checksum is produced from the message M as follows:

$$\text{checksum} = \mathrm{M}[1] \oplus \mathrm{M}[2] \oplus \ldots \oplus \mathrm{Y}[\mathrm{m}] \oplus \mathrm{C}[\mathrm{m}]0^*$$

where C[m]0* consists of C[m] padded with least significant bits to the length n. Finally, an authentication tag of length τ is generated, using the same key as is used for encryption:

$$\text{tag} = \text{first } \tau \text{ bits of } \mathrm{E}_K(\text{checksum} \oplus \mathrm{Z}[m])$$

algorithm OCB-Encrypt$_K$(N, M)
Partition M into M[1]…M[m]
L ← L(0) ← $E_K(0^n)$
R ← E_K(N ⊕ L)
for i ← 1 **to** m **do** L(i) ← 2 · L(i −1)
L(−1) = L · 2^{-1}
Z[1] ← L ⊕ R
for i ← 2 **to** m **do** Z[i] ← Z[i − 1] ⊕ L(ntz(i))
for i ← 1 **to** m − 1 **do**
  C[i] ← E_K(M[i] ⊕ Z[i]) ⊕ Z[i]
X[m] ← len(M[m]) ⊕ L(−1) ⊕ Z[m]
Y[m] ← E_K(X[m])
C[m] ← M[m] ⊕ (first len(M[m]) bits of Y[m])
Checksum ←
  M[1] ⊕ … ⊕ M[m −1] ⊕ C[m]0* ⊕ Y[m]
Tag ← E_K(Checksum ⊕ Z[m]) [first τ bits]

algorithm OCB-Decrypt$_K$(N, M)
Partition M into M[1]…M[m]
L ← L(0) ← $E_K(0^n)$
R ← E_K(N ⊕ L)
for i ← 1 **to** m **do** L(i) ← 2 · L(i −1)
L(−1) = L · 2^{-1}
Z[1] ← L ⊕ R
for i ← 2 **to** m **do** Z[i] ← Z[i − 1] ⊕ L(ntz(i))
for i ← 1 **to** m – 1 **do**
  M[i] ← D_K(C[i] ⊕ Z[i]) ⊕ Z[i]
X[m] ← len(M[m]) ⊕ L(−1) ⊕ Z[m]
Y[m] ← E_K(X[m])
M[m] ← (first len(C[m]) bits of Y[m]) ⊕ C[m]
Checksum ←
  M[1] ⊕ … ⊕ M[m − 1] ⊕ C[m]0* ⊕ Y[m]
Tag' ← E_K(Checksum ⊕ Z[m]) [first τ bits]

Figure 23.7 OCB Algorithms

The bit length τ of the tag varies according to the application. The size of the tag controls the level of authentication. To verify the authentication tag, the decryptor can recompute the checksum, then recompute the tag, and finally check that is equal to the one that was sent. If the ciphertext passes the test, then OCB produces the plaintext normally.

Figure 23.7 summarizes the OCB algorithms for encryption and decryption. It is easy to see that decryption is the inverse of encryption. We have

$$\begin{aligned} E_K(M[i] \oplus Z[i]) \oplus Z[i] &= C[i] \\ E_K(M[i] \oplus Z[i]) &= C[i] \oplus Z[i] \\ D_K(E_K(M[i] \oplus Z[i])) &= D_K(C[i] \oplus Z[i]) \\ M[i] \oplus Z[i] &= D_K(C[i] \oplus Z[i]) \\ M[i] &= D_K(C[i] \oplus Z[i]) \oplus Z[i] \end{aligned}$$

23.4 KEY TERMS AND REVIEW QUESTIONS

Key Terms

actuator	edge	microcontroller
backbone network	fog	operational technology (OT)
cloud	information technology (IT)	sensor
core	Internet of Things (IoT)	transceiver

Review Questions

23.1 Define the Internet of Things (IoT).

23.2 List and briefly define the principal components of an IoT-enabled thing.

23.3 Define the patching vulnerability.

23.4 Define tamper resistance and tamper detection.

23.5 What is MiniSec?