CHAPTER 4

IoT Node Authentication

Shancang Li

The IoT aims at enabling a number of next generation technologies, such as intelligent wireless sensor networks (WSNs), smart cities, smart homes, and mobile-health (m-health) systems. These scenarios require secured solutions to prevent leakage of private information and harmful actuating activities by means of peer authentication and secure data transmission between the IoT nodes and servers. However, the existing IP-based IoT structure and primitives are not fully designed with the limitation of resource-constrained IoT devices (such as energy consumption, computation resource, communication ranges, RAM, FLASH, etc.). As a result, more lightweight security solutions are necessary to ensure the security at resource-constrained IoT devices.

In IoT environment, the limitation at IoT end-nodes includes following aspects:

- Processing power, CPU(MCU) processor, RAM
- Storage space
- Network capacity
- Lack of user interface and display
- Energy consumption

In this chapter, we will discuss the following commonly used security protection technologies in constrained IoT environment:

- Security goals in IoT
- Public-key-based authentication
- Identify-based authentication, encryption, and digital signature
- Lightweight cryptography primitives in IoT
- Secure enabling techniques for resource-constrained IoT
- Existing security solutions in IoT

Securing the Internet of Things. DOI: http://dx.doi.org/10.1016/B978-0-12-804458-2.00004-4

4.1 SECURITY GOALS IN IoT

Similar to existing IP networks, in the different scenarios of IoT, the cryptographic primitives in IoT are utilized to comply with the main security goals for exchanged message and the system itself. The basic security goals in IoT are:

1. Confidentiality: The message is only disclosed to authorized entities, user, nodes, devices, and services; the confidentiality is about the controlling for devices, message access. The private data, keys, and security credentials must be well protected from unauthorized entities.
2. Integrity, the original message is not tampered with: In IoT systems, different applications may have various integrity requirements, such as e-healthcare system may have more restricted data integrity than the general smart cities applications.
3. Authentication and authorization: The connectivity of the devices aggravates the problem of authentication because of the access control and the nature of wireless communication in IoT systems.
4. Availability: The system keeps serving its purpose and stays uninterruptedly available for legitimate entities. The IoT systems are required to be robust to provide services for accessing anytime.
5. Accountability: To improve the robustness of services in IoT environment, accountability of IoT systems is necessary.

Attack techniques in IoT environment are important to understand:

1. Physical attacks, which means attack tampers with physical components. In some case, the IoT devices might be deployed in outdoor environment, which brings risks to IoT systems.
2. Eavesdropping is the process of overhearing an ongoing communication, which is as well preliminary for launching the next two attacks. Since in IoT environment, many IoT end-nodes are interconnected wirelessly and everyone is able to access the medium. Confidentiality is a typical counter-measurement against eavesdroppers. However, if the keying material is not exchanged in a secure manner, the eavesdropper could be able to compromise the confidentiality. Therefore, secure key change algorithms, such as DH (Diffie-Hellman), are used in the practical scenario.
3. Impersonation is when a malicious entity pretends to be another, mostly legitimate, entity, for instance will be replaying a genuine message, in order to bypass the aforementioned security goals. A special form of this attack is the man-in-the-middle (MITM) attack.
4. MITM attack takes place when a malicious entity is on the network path of two genuine entities. Hence, it is capable of delaying, modifying, or dropping messages. MITM attack is interesting within the context of public-key cryptography (PKC). Then the malicious entity

does not attempt to break the keys of involved parties, but rather to become the falsely trusted MITM. The malicious user achieves this by replacing the exchanged keys with its own. This way each of the parties establishes a secure channel with the malicious user, who gains access to messages in plain text.

5. DoS (Denial of Service) attack targets the availability of a system that offers services. This is achieved by exhaustingly consuming resources at the victim so that the offered services become unavailable to legitimate entities. A common way to launch this attack is to trigger expensive operations at the victim that consume resources, such as computational power, memory, bandwidth, or energy. This attack is critical for constrained devices, where existing resources are already scarce.
6. Access attacks that involve attacks unauthorized entities gain access to IoT systems or devices.
7. Other attacks, such as firmware attack as "bad USB," attacks on privacy, RAM attacks, channel side attack, ransomware, etc.

4.2 PUBLIC-KEY-BASED AUTHENTICATION

In IoT, authentication is the process of identifying users, devices, applications, and restricting access to authorized users and nonmanipulated devices or services. In this process, the username and password-based cryptographic schemes are used to provide a robust secure operation over the IoT. The authentication mechanisms can provide the IoT following benefits:

- Robust devices and secure communication for users
- Development of new services over IoT
- Avoidance of embarrassing data breaches
- Strong anticounterfeiting and antitampering capability
- Reduce risk of third-party services

The public-key-based authentication is widely used in current Internet; however, it is impracticable for constrained environment such as IoT due to expensive cryptographic operations. In this section, we will investigate public-key-based authentication and analysis how to tailor it for light cryptographic in constrained IoT environment. The authentication of IoT end-nodes is an important issue to provide basic secure protection of the network and devices. The node authentication in IoT involves the following:

- Smart objects, small device with specific purpose, low cost, limited abilities;
- IoT, interconnect things and their users to enable new applications;
- IoT nodes are expected to be integrated in all aspects of existing works, entrusted with vast amounts of data, need to communicate unseen and autonomously.

Table 4.1 Resources Classification for IoT Nodes

Name	Data Size (e.g., RAM)	Code Size (e.g., Flash)
Class 0, C0	≪10 KiB	≪100 KiB
Class 1, C1	~10 KiB	~100 KiB
Class 2, C2	~50 KiB	~250 KiB

Existing RFC7228: Terminology for IoT node networks (constrained environment)

- Device classification
- Energy profile
- Sleep strategies

Table 4.1 shows the resources classification for IoT end-nodes.

Cryptography is widely used in networks to protect private communications and a number of ciphers have been developed, such as Data Encryption Standard, the Ron Rivest, Adi Shamir and Leonard Adleman (RSA) was the first practical public-key cryptosystem.

In IoT environment, the communications between nodes and the infrastructure node require light key distribution method using the public key to reduce the burden; however, it is difficult to apply the method to a light device where the encryption module, such as advanced encryption standard (AES), RSA, elliptical curve cryptography (ECC), cannot be mounted. The **internet engineering task force** (IETF) is considering application of transport layer security (TLS), datagram transport layer security (DTLS), IPSec, etc., which have been adopted in the IP-based networks. The basic concept is to apply DTLS to constrained application protocol (CoAP), which is the key protocol in IoT. In this section, we will review the basic concepts of public-key schemes, symmetric cryptography, and its application in encryption. Then, we cover the PKC and public-key infrastructure (PKI), specifically with regard to X.509 certificates and RAW Public Keys (RPKs).

The basic goals of an authenticated authorization protocol in IoT include:

- Secure exchange of authorization information
- Establish DTLS channel between constrained nodes
- Use only symmetric key cryptography on constrained nodes
- Support of class-1 devices
- RESTful architectural style
- Relieve constrained nodes from managing authentication and authorization

The authenticated authorization

- Determine if the owner of an item of interest allows an entity to access this item as requested.
- Authentication: Verify that an entity has certain attributes (cf. RFC4949).
- Authorization: Grant permission to an entity to access an item of interest.
- Authenticated Authorization: Use the verified attributes to determine if an entity is authorized.

4.2.1 Symmetric Cryptography

A symmetric-key system is used to provide confidentiality of message in transmission, storing, and processing. The symmetric-key algorithm performs the operations of encryption/decryption based on a single key that is shared by two or more parties. A difficulty in symmetric cryptography is securely delivering the key from the encoder to the decoder(s) can introduce a security risk. Anyone who gains access to the symmetric key is able to access/modify/send the message without the recipient's knowledge that the message has been modified. To fix these issues, public-key cryptography or asymmetric key have been developed. The symmetric cryptography algorithms are usually grouped into stream ciphers and block ciphers. The AES is a commonly used block cipher encryption algorithm in network security solutions.

In symmetric-key encryption, the secret key K, the plain text message P, and the cipher text C have the same length. For example, in AES 128, the length of K, P, C are all 128 bits (16 bytes), both the encryption and decryption operations consist of XORing, permutations, bit-shifting, and linear mixing functions that are performed in a known order. In general, the original plain text is divided into multiple blocks with fixed length:

$$C_i = \text{Encrypt}\,(K,\ P_i), \quad \forall i = 1, \ldots, n$$

The weakness of this is that the same plain text blocks result in same cipher blocks. This is especially critical for packets with a known format and a repeating pattern in the content. To introduce randomness into cipher blocks and make decryption attacks difficult, Cipher Block Chaining (CBC) can be used where before encryption each plain text block is Exclusive operation (XORed) with the previous cipher block.

In Fig. 4.1, the first cipher block C_0, which is XORed to the first plain text block as input, is referred to as the initialization vector (IV). Except the IV, all the following cipher blocks are dependent on all the previous cipher blocks due to the XORing. This feature is used in CBC-message authentication code (MAC) to provide authentication and integrity protection. In Fig. 4.1, the last cipher block Cn serves as the MAC.

$$C_i = \text{Encrypt}\,(K,\ P_i \oplus C_{i-1}), \quad \forall i = 1, \ldots, n,\ C_i = \text{IV}$$

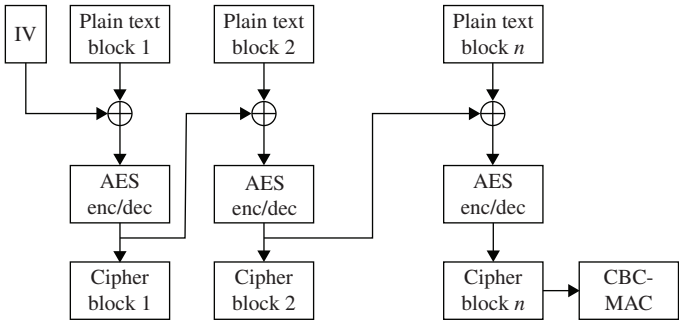

FIGURE 4.1
AES block encryption.

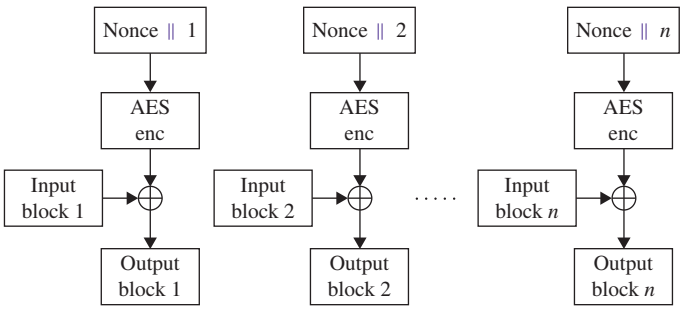

FIGURE 4.2
AES-CTR block encryption.

A MAC provides information that allows to authenticate a message and to verify the integrity of it. Practically, a more typical way than CBC-MAC is used to create the MAC of a message M is by using a hash function with a shared secret key K:

$$\text{MAC}(M) = \text{HASH}(K|M) = D$$

A secure cryptographic hash function generates from a variable input, a fixed length of output.

The AES-CTR is another block cipher encryption algorithm that, in contrast to CBC, uses a Nonce and a counter to add randomness to each cipher block, as shown in Fig. 4.2.

The input can be a plain text or cipher block and the output is the corresponding cipher or plain text block, respectively.

$$K_i = \text{Encrypt}\,(K,\ \text{Nonce}||i\,), \quad \forall i = 1, \ldots, n$$
$$C_i = P_i \oplus K_i$$

The decryption in counter (CTR) mode is performed in the same fashion as encryption which utilizes the following feature of XOR:

$C_i \oplus K_i = P_i \oplus K_i \oplus K_i = P_i$

As a result, CTR does not use AES decryption.

4.2.1.1 AES-CCM

AES-CCM is a mode of operation for block ciphers, which is developed to provide at the same time confidentiality, authentication, and integrity protection. This is achieved by encryption in CTR mode and creating the CBC-MAC of the input. The CBC-MAC is 128 bits but can be truncated to any length. It is then appended to the end of the cipher text. Since CBC-MAC and CTR are performed into separate steps, there is the possibility of selectively not encrypting the entire input, but integrity protecting it entirely. This feature puts **counter with CBC-MAC** (CCM) in the class of algorithm that provides authenticated encryption with associated data.

Since the AES-CCM only relies on AES encryption, most of IoT chips have a hardware built-in AES engine. This makes AES-CCM the favorable choice of encryption for constrained devices or sensors. In IoT, the standardization community requires AES-CCM as the mandatory cipher suite for DTLS in secure CoAP.

4.2.2 Public-Key Cryptography

The symmetric key algorithms are quite efficient, but the key distribution is difficult to IoT end devices. The key distribution requires a secure connection between the key distribution sever and the IoT nodes. PKC and asymmetric cryptography are two effective ways of providing confidentiality and authentication. In contrast to the symmetric cryptography, the PKC is based on mathematically hard problem to solve, whereas hard in this context refers to the complexity of calculation. The public-key encryption is based on "trapdoor" functions, which are easy to compute, but hard to reverse without additional information. The RSA is a widely used public-key algorithm, in which the hard problem is finding the prime factors of a composite number. In PKC cryptosystem, generally in a key pair, the public key and the private key, the public key is made accessible to the public and the private key is kept at a safe place. The public key is generally used in two ways.

1. Public-key encryption, in which one is capable to encrypt a message with the public key of an entity, where only the entity with the corresponding private key is capable of decrypting the cipher text.
2. Digital signatures, in which a cipher text generated with the private key can be decrypted by anyone who has the public key. This verification proves that the sender had access to the private key and therefore is likely to be the person associated with the public key.

Table 4.2 Key Size for Symmetric Key, RSA, and ECC

Symmetric Key	RSA Key	Elliptic Curve Key
80	1024	160
112	2048	224
128	**3072**	**256**
192	7680	384
256	15,360	521

In PKC system, public/private key pairs can be easily generated for encryption and decryption. The security strength in a PKC system lies in how difficult to determine a properly generated private key from its public key. In this case, the length of private key is important for avoiding brute-force attacks.

The RSA is one of the first practical public-key cryptosystems, which is based on the practical difficulty of factoring the product of two large prime numbers. If the public key is large enough, only the one knowing the prime numbers can feasibly decode the message. The RSA is a relative slow algorithm for encryption however it is commonly used to pass encrypted shared keys for symmetric key cryptography. Since RSA encryption is an expensive operation, in IoT it is rather used in combination with symmetric cryptography. The shared symmetric key is encrypted with RSA; the security of encryption in general is dependent on the length of the key. For RSA, a key length of 1024 bits (128 bytes) is required, to have an equivalent security level of symmetric key cryptography with a key length of 128 bit (16 bytes). The large key size of RSA will cause expensive computation costs.

The ECC is an alternative to common PKC because of the resistance against powerful index-calculus attacks. The ECC allows efficient implementation due to a significant smaller bit size of the operands over resource-constrained environment. ECC is another public-key cryptography approach that works based on elliptic curves over finite fields. ECC's smaller key size is 256 as shown in Table 4.2. It is more efficient than RSA and it is more suitable for resource-limited devices in IoT. The basic idea of ECC is the general assumption that the elliptic curve discrete logarithm problem is infeasible or at least not solvable in a reasonable time.

4.2.3 Public Key Infrastructure

A PKI is a set of roles, policies, and procedures needed to create manage, distribute, use, store, and revoke digital certificates and manage public-key

encryption (Wiki). In IoT environment, the general public-key problem is the requirement of an authenticated exchange of public keys. The PKI consists of components to securely distribute public keys and is today widely used in the traditional Internet. The most important PKI is a trusted third party who signs the identifier of an entity with its private key.

Interconnected devices in IoT environment must provide trustworthy information to users and services; however, establishing trust across large-scale network is a significant challenge. The devices in IoT are easily attacked and the communications between nodes in IoT are usually difficult to secure. The PKI system works well in existing systems such as banking systems, cellular stations, mobile networks, and are proven to be able to provide trusted environment. So the PKI is a promising solution in IoT.

- PKI comes to assurance and validation.
- Scale. The PKI deployments certainly exist that have the ability to manage millions of certificates, most operate at significantly smaller level.
- Technology issue. Extremely low-power and low-budget device will populate the IoT. Traditional cryptography is not designed for these environments and is mathematically intensive, which requires CPU power. Another problem is credential generation. Making good keys is not easy, and making them in high volumes can quickly become a bottleneck. Again, cryptoalgorithms designed for low-power devices and rapid key generation already exist and have been widely proven.

Before we detail how the PKI works, we first introduce the basic concepts in PKI. The trusted third party is referred to as certification authority (CA) who issues a certificate which mainly constrains the public key and the identifier of an entity. The main elements include:

- Subject: the identifier of the entity whose public key is being certified
- Signature: the algorithm used to create the signature
- Subject PKI: subjects public key and identifier of the algorithm used to generate
- Validity: the time period the certificate can be used
- Issuer: CA's identifier
- Signature value: The issuer's signature on the hash of the previous elements.

Access to the public key of the CA is required to verify the certificate. This brings us in the original problem. Root CAs are at the highest level of trusted hierarchy and have self-signed certificates. Furthermore, root CAs are predeployed into systems for instance via browser vendors.

4.3 IDENTIFY-BASED AUTHENTICATION, ENCRYPTION, AND DIGITAL SIGNATURE

4.3.1 Identify-Based Authentication

Technically, IoT consists of uncountable devices, sensors, or actuators or simply objectives connected to services in the Internet. These objectives are from different vendors, communities, or standard groups. Most of these devices speak different protocols, which make the IoT hard to be implemented. In this case, the devices identify management as one of the most important common technologies, which should be able to coordinate different protocols, standards, scenarios. From a security point of view, security protection should be provided for "Identities of things" in heterogeneous communication and machine-to-machine security. The security challenges are related to identification, authentication, privacy, trustworthiness, and confidentiality. The identification is one of the most important challenges in security of IoT. IoT consists of variety of smart devices like intelligent sensors, smart objectives, computer, back-bone servers, cloud clusters, etc. All of them should be uniquely identified for addressing capabilities and for providing a means to communicate with each other. From the viewpoint of security, the security protection mechanism should be able to identify the message generators, transmitters, and receivers. Existing identification schemes, for example, RFID objective identifier, EPC global, NFC, IPv4, IPv6, etc., have been developed for existing networks, however, how to securely manage devices in IoT environment is still a challenge.

The commonly used protocols for identity authentication include:

- One-way authentication, which authenticates two nodes. For example, node 1 and node 2 have a common secret key X_{uh}. Node selects $r \in GF(P)$ which will be used to create session key. T_u is time stamp of nodes. The secret key created by node 1 is $L = h(X_{uh} \oplus T_u)$, then node 1 encrypts r with L as $R = E_L$ and encrypts T_u with X_{uh} as $T_{us} = EX_{uh}(T_u)$. $MAC_1 = MAC(X_{uh}, R||ICAP_1)$, where $ICAP_1$ is a data structure represented by an identity based on node 1. Now, node 1 sends the following parameters to node 2 (R, T_{us}, MAC_1). Node 2 generates its time stamp as $T_{current}$ and decrypt T_{us} to get T_u and compare it with $T_{current}$. If $T_{current} > T_u$, it is valid. Now in node 2 calculate L and decrypt R to get r. It also calculates MAC'_1 and it will verify this with MAC_1 received from node 1. Fig. 4.3 shows the protocol.
- Mutual-authentication, which is part of authentication authenticates node 2 to node 1. Node 2 builds a MAC as $MAC_2 = MAC(r||ICAP_2)$ and also encrypts r with X_{uh} as $R' = EX_{uh}(r)$. Then it sends (R', MAC_2) to

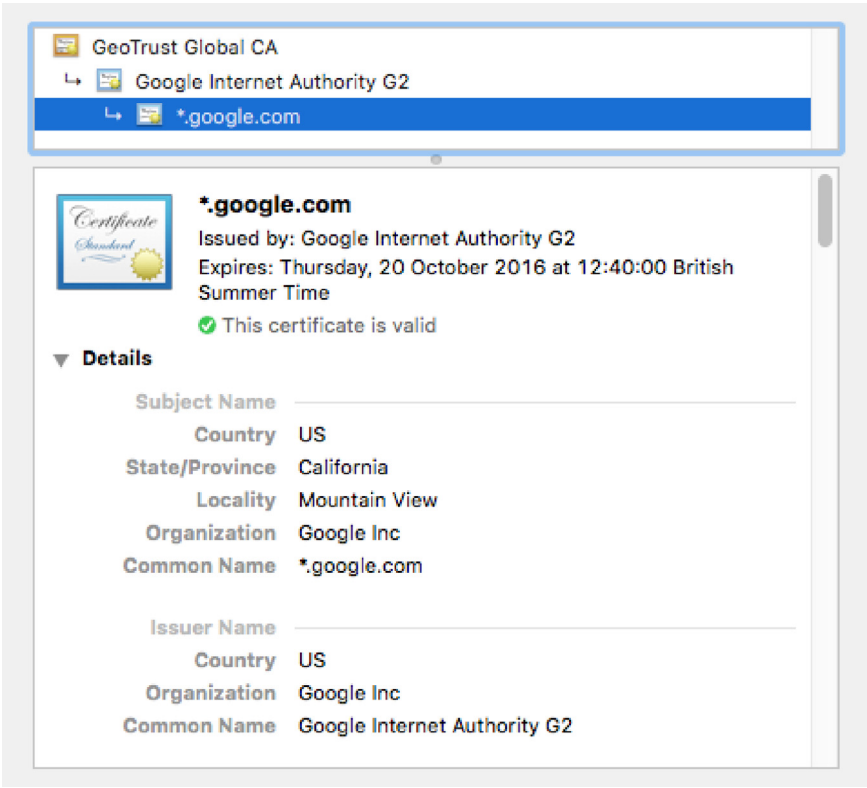

FIGURE 4.3
Example of Google CA.

node 1. Node 1 verifies MAC_2 and decrypts R' and compares received r with this r'. Fig. 4.4 shows the protocol.

Two of the best-known uses of PKC are:

- Public-key *encryption*, a message is encrypted with a recipient's public key. The message can only be decrypted by the matching private key, who is assumed to be the owner of the key and the person associated with the public key. This is used in an attempt to ensure confidentiality.
- Digital signatures. A message is signed with the sender's private key and can be verified by anyone who has access to the sender's public key. This verification provides that the sender had access to the private key, and therefore is likely to be the person associated with the public key. This ensures that the message has not been tampered with.

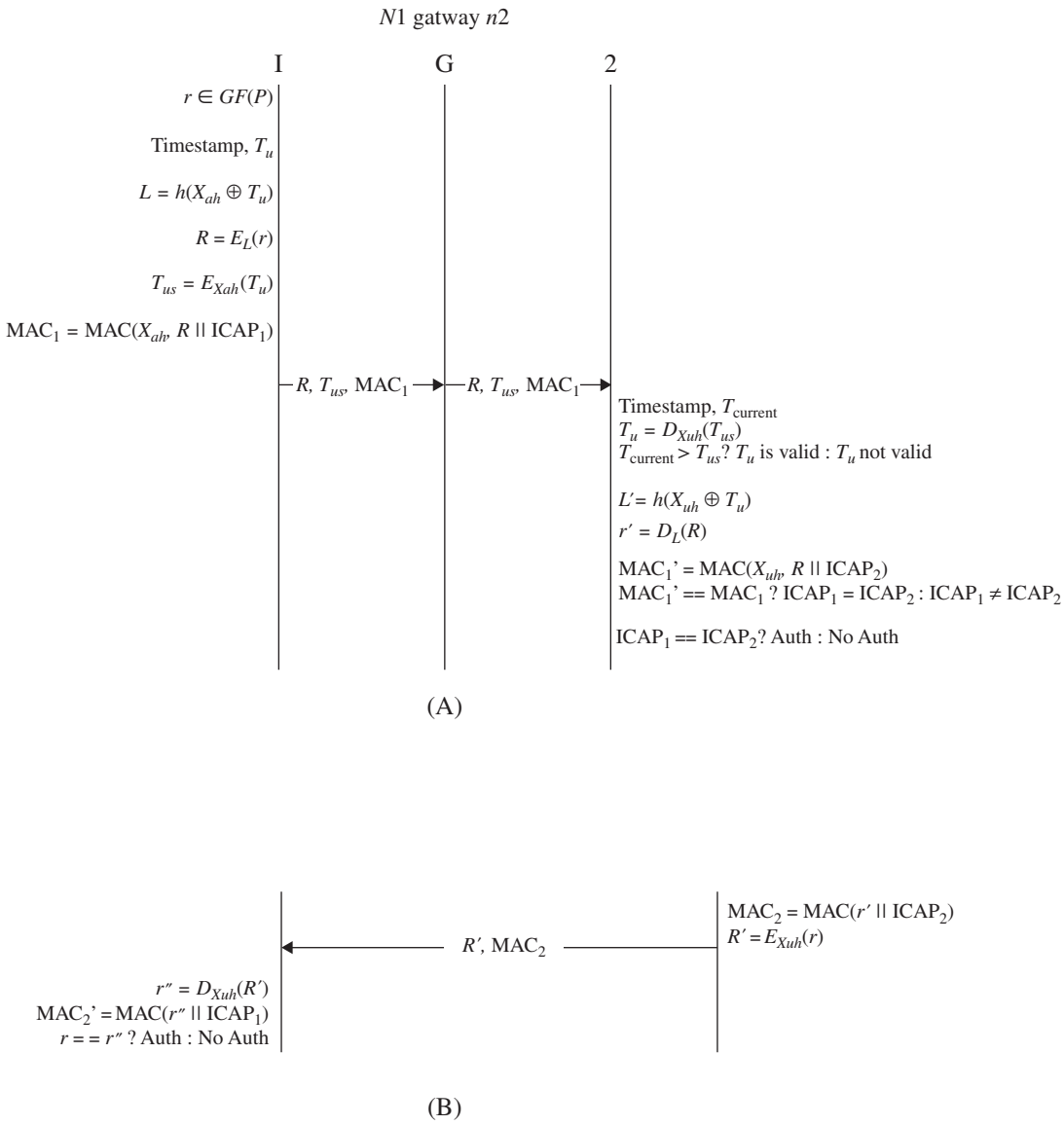

FIGURE 4.4
(A) One-way authentication and (B) mutual authentication.

4.3.2 Digital Signature

A problem with the use of public-key cryptography is confidence/proof that a particular public key is authentic. It is correct and belongs to the person or entity claimed, and has not been tampered with or replaced by a malicious third party.

The usual approach to the problem is to sue PKI, in which one or more third parties—known as CAs—certify ownership of key pairs. To date, no fully satisfactory solution to the "public-key authentication problem" has been found.

The symmetric key algorithms are quite efficient, but the key distribution is difficult at IoT end devices. The key distribution requires a secure connection between the key distribution sever and the IoT nodes. PKC and asymmetric cryptography are two effective ways of providing confidentiality and authentication. In contrast to the symmetric cryptography, the PKC is based on mathematically hard problem to solve, whereas hard in this context refers to the complexity of calculation. The public-key encryption is based on "trapdoor" functions, which are easy to compute, but hard to reverse without additional information. The RSA is a widely used public-key algorithm, in which the hard problem is finding the prime factors of a composite number. In PKC cryptosystem, generally is a key pair, the public key and the private key, the public key is made accessible to the public and the private key is kept at a safe place. The public keys are generally used in two ways.

1. Public-key encryption, in which one is capable of encrypting a message with the public key of an entity, where only the entity with the corresponding private key is capable of decrypting the cipher text.
2. Digital signatures in which a cipher text generated with the private key can be decrypted by anyone who has the public key. This verification proves that the sender had access to the private key and therefore is likely to be the person associated with the public key.

In PKC system, public/private key pairs can be easily generated for encryption and decryption. The security strength in a PKC system lies in how difficult to determine a properly generated private key from its public key. In this case, the length of private key is important for avoiding brute-force attacks.

The RSA is one of the first practical public-key cryptosystems, which is based on the practical difficulty of factoring the product of two large prime numbers. If the public key is large enough, only the one knowing the prime numbers can feasibly decode the message. The RSA is a relative slow algorithm for encryption; however, it is commonly used to pass encrypted shared keys for symmetric key cryptography. Since RSA encryption is an expensive operation, in IoT it is rather used in combination with symmetric cryptography. The shared symmetric key is encrypted with RSA, the security of encryption in general is dependent on the length of the key. For RSA, a key length of 1024 bits (128 bytes) is required, to have an equivalent security level of symmetric key cryptography with a key length of 128 bit (16 bytes). The large key size of RSA will cause expensive computation costs.

The ECC is an alternative to common PKC because of the resistance against powerful index-calculus attacks. The ECC allows efficient implementation

due to a significant smaller bit size of the operands over resource-constrained environment. ECC is another public-key cryptography approach that works based on elliptic curves over finite fields. ECC's smaller key size is 256 as shown in Table 4.2. It is more efficient than RSA and it is more suitable for resource-limited devices in IoT. The basic idea of ECC is the general assumption that the elliptic curve discrete logarithm problem is infeasible or at least not solvable in a reasonable time.

The IETF recommends the AES-CCM in combination with ECC for constrained devices. In this section, we will explain how ECC is used to perform a secure key exchange and create digital signatures.

- ECC concept
- Secure key exchange
- Digital signature

The equation of an elliptic curve has the following form:

$$y^2 = x^3 + ax + b$$

The set of EC points are on this curve. A feature of EC is that the result of addition of two points on the curve lies again on the curve. The same holds as well for multiplication. Assume P is a known point on a given EC, and d is a secret random number which serves as the private key, the public key Q, and the private key d have the following relation:

$$Q = d \times P$$

Then, the public key Q is again a point on the same curve. Although Q and P are publicly known and Q is the result of adding P and d times to itself, it is mathematically a hard problem to compute d.

Public keys are created by multiplying the generator. Using the routines for arithmetic, other routines can be built that will compute scalar multiples of the generating point, kP, or of other points $Q = dG$. Public keys are created by multiplying the generator, that is Q is the public key for d if $Q = dP$ on the elliptic curve. Key generation is the production of (d, D) is therefore very basic and efficient in ECC. In RSA key generation involves coming up with large prime numbers and takes much longer.

Assume user Q wants to sign a message m, he/she first computes $K = kP$ for k random, since this can be complete before the message is in hand, so it is often completed over powerful service and passed to the constrained nodes in IoT. If the message m can be signed by computing with much less intensive modular computations over nodes:

$$r = x_{\text{coord}}(K = kP) \bmod n$$

$$s = K^{(-1)}(m + dr)$$

in which n is the pointer order and the signature on message m is (r, s). If one knows the public key D, then he can verify this signature on m as:

$$K' = (s^{n-1}m)P + (s^{(-1)}r)Q$$

$$r' = x_{\text{coord}}(K')$$

if the r and r' are the same, it means it is acceptable. In practical, the applications that require cryptography system can quickly generate signatures and a number of speeding up verification based on ECC have been developed.

The ECC has small key sizes and is able to generate efficient signature. The strength and efficiency of ECC makes it an ideal for many IoT applications over resource-limited devices. The ECC is suitable for securing IoT environment where more resource-constrained devices are interconnected, such as intelligent sensors, wireless sensor nodes, and e-healthcare devices.

4.3.3 Raw Public Key

In resource-constrained IoT devices, such as intelligent sensors or RFID tag, the certificate chains or even single certificate may be too big to process. Recently, the RPKs are recommended by IEFT instead of the certificates for TLS and DTLS. The RPK requires the out-of-band validation of the public key:

1. Obtaining the public key via DNS-based authentication of named entities or authentication via DNS security extensions
2. Predeployment of RPKs is beneficial in IoT-constrained devices which are configured before deployment with the public key of the back-end service.

The RPK contains the subject Public-Key Information of a certificate which carries the public key values and the algorithm identifier of the cryptographic algorithm used to generate it. RPKs allow for omitting large certificates from the handshake; however, it requires an out-of-band technique for the verification of the public key.

It should be noted that if an IoT gateway node supports the RPK certificates, it must support specific cipher suites such as TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 (CoAP) and TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256. The end IoT nodes must support at least one of the above cipher suites. The client node uses the value of the "Public Key or Identify" resource for its RPK certificate to determine the expected value of the server's RPK and the value "Secret Key" resource for its private key. The client must check whether the RPK presented by the server exactly matches with the stored public key. The RPK mode is appropriate for IoT nodes deployments where

there is an existing trust relationship between the client and server. The server must store its own private and public keys, and must have a stored copy of the expected client public key. The server must check that the RPK present by the IoT client exactly matches with the stored public key. In some application scenarios, such as smartcard, the RPK certificates provisioning needs no preexisting trust relationship between server and client. The preestablished trust relationship is simply between the server and the smartcards.

4.3.4 X.509 Certificates

X.509 is an important standard in cryptography, which is designed for a PKI to manage digital certificates and public-key encryption. The X.509 is a key part of the TLS and it is widely used in web, mobile, and email security. In X.509, an organization that needs a signed certificate requests one via a certificate signing request (CSR). To do this (1) they first generate a key pair, keeping the private key secret and using it to sign the CSR, which contains the public key that is used to verify the signature of the CSR and the distinguished name (DN) and (2) the certification authority issues a certificate binding a public key to particular DN.

The Firefox, Chrome, Safari, etc. come with a predetermined set of root certificates preinstalled, so SSL certificates from large vendors will work instantly. In effect the browsers' developer determine which CAs are trusted third parties for the browser's users (Fig. 4.5).

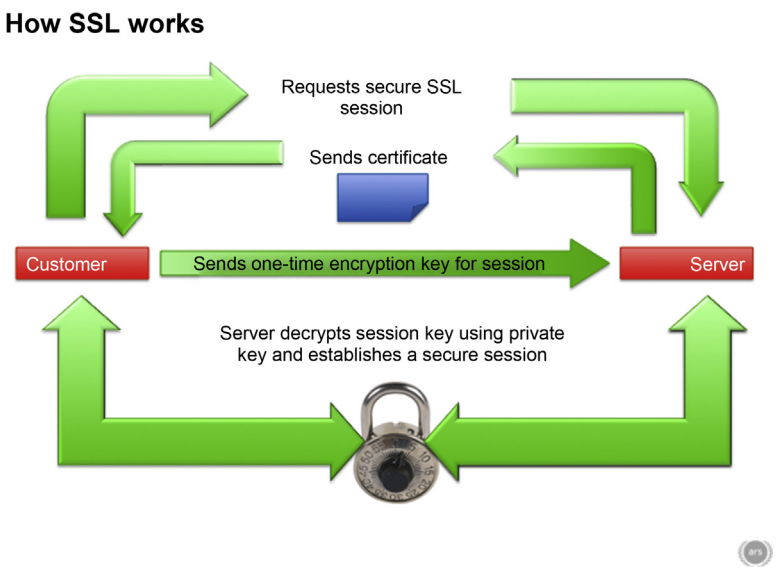

FIGURE 4.5
AES-CTR block encryption.

X.509 certificates are the dominating type of certificates and are consequently used in the certificate-based model of DTLS. In this section, we briefly address the concepts of X.509.

The X.509 certificates are encoded into Base64 which is a binary-to-text encoding scheme. The basic structure is

- *Identifier*
- *Length*
- *Content*

4.4 IP CONNECTIVITY

IoT is a hybrid network that contains different networks: WSNs, Mobile networks, IP, and wireless mesh networks. Most existing IoT solutions are undergoing the IP-enabled and thus connected to the Internet. As a result, existing and matured IP-based security protocol is within constrained environment. Since the existing IP-based security protocol is not designed for resource-constrained devices, such as intelligent sensors, it cannot be used just directly in IoT. It is needed to redesign the existing IP-based protocols or improve it for IoT devices. The TLS is the underlying security protocol for applications protocols, such as HTTP, HTTPS, and it runs over TCP. In IoT, the UDP has become the de facto favorable protocol since it is simple and efficient. The CoAP is intended to be used in resource-constrained devices and widely used in IoT and machine-to-machine networks.

Fig. 4.6 shows protocols that have been developed at different layers of IoT, including messaging protocols at application layer, such as CoAP, routing protocols (such as the routing protocol for low power and lossy network, RPL). In this protocol, the IPv6 is one of the most important enablers in the IoT environment that supports the possibility to connect billions of smart objectives together. However, all protocols should be designed by following the security requirements.

Communication in IoT-constrained environment

- CoAP (RFC 7252), which is designed for special requirements of constrained environments like IoT and similar to HTTP with RESTful architecture style
- DTLS binding
- User controls the device and data through authorization

4.4.1 Datagram Transport Layer Security

In the Internet, the TLS is a prominent IP-based security protocol which is widely used to provide protection over transparent connection-orient channel

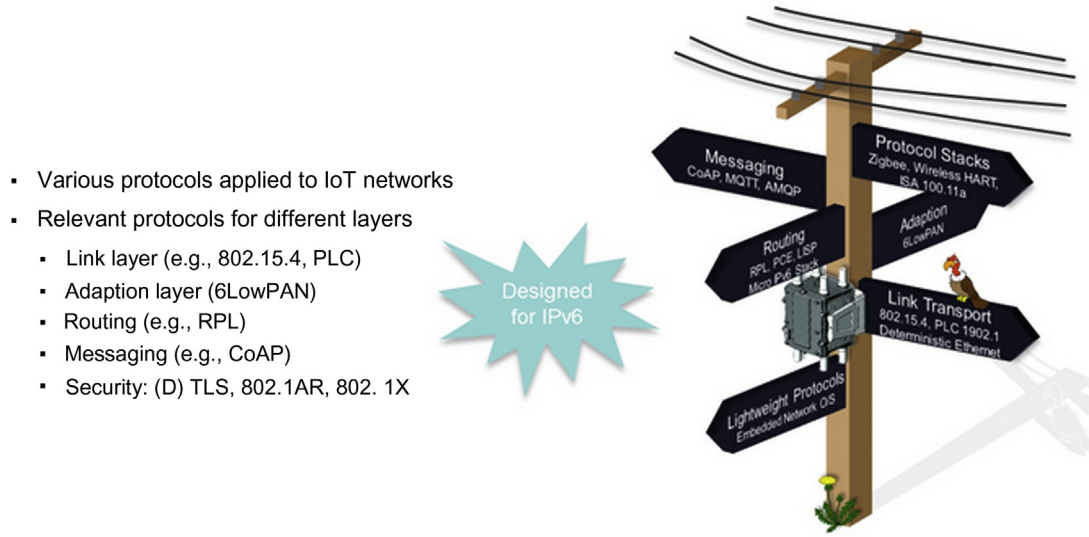

FIGURE 4.6
Protocols in IoT.

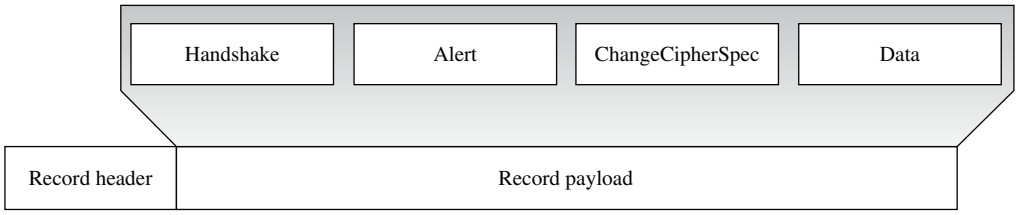

FIGURE 4.7
Structure of the DTLS.

against security attacks, such as eavesdropping, tampering, or message forgery. In web applications, the TLS is widely used for web protocols, such as HTTP and TCP. Fig. 4.7 shows the structure of DTLS.

In IoT applications, the security protocol is particularly targeted for small, low-power sensors, switches, valves, and similar components that need to be controlled or supervised remotely, through standard Internet works. The DTLS is developed based on TLS by providing equivalent security services, such as confidentiality, authentication, and integrity protection. The TLS uses the TCP and therefore does not encounter packet reordering and packet loss issues. In DTLS, a handshake mechanism is designed to deal with the packet-loss, reordering, and retransmission. In DTLS, the initial authentication of the peers and key agreement and then data protection is provided via the

secure channel. In DTLS, the lower layer is the record protocol which protects all DTLS messages as shown in Fig. 4.7. The upper layer is record protocol payload; it consists of four protocol types:

- Handshake, DTLS provides three types of handshake: nonauthentication, server authentication, and server and client authentication
- Alert
- ChangeCipherSpec
- Data

Mutual certificate-based DTLS handshake. Client and server possess a pair of private–public keys. They exchange during the handshake their public keys. Each public key is bound to an identity by means of a certificate. For freshness of keying material and providing perfect forward secrecy random values and ephemeral DH key pairs are generated at each side, exchanged and incorporated into the calculation of the keying material (Fig. 4.8).

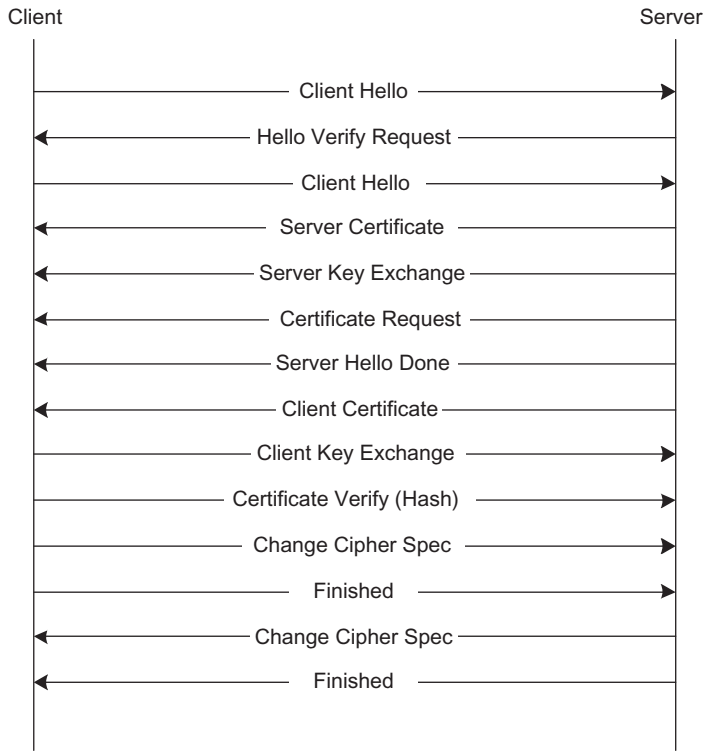

FIGURE 4.8
Mutual certificate-based DTLS handshake procedure.

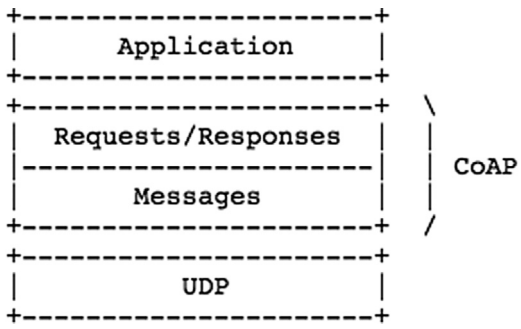

FIGURE 4.9
Structure of the CoAP.

4.4.2 Constrained Application Protocol

The CoAP is particularly designed web transfer protocol for use with resource-constrained networks and devices. It is very suitable for IoT environment, where lots of end-nodes often have only 8/16-bit microcontrollers with small amounts of ROM and RAM, while constrained network such as IPv6 over low-power Wireless Personal Area Networks (6LoWPANs) often have high packet error rates and a typical throughput of 10 s of kbps. The CoAP provides a request–response interaction model between applications. CoAP supports built-in discovery of services and resources, includes key concepts of the Web, URIs, etc. Fig. 4.9 shows the basic structure of CoAP.

CoAP defines four types of message:

- Confirmable
- Nonconfirmable
- Acknowledgment
- Reset

The basic exchange of the four types of messages are somewhat orthogonal to the request–response interactions; requests can be carried in confirmable and nonconfirmable message, and responses can be carried in these as well as piggybacked in acknowledge messages.

4.5 LIGHTWEIGHT CRYPTOGRAPHY

We propose to adopt new advancing technology, "Lightweight Cryptography," in the IoT. We described two reasons that support our proposal.

4.5.1 Efficiency of End-to-End Communication

In order to achieve end-to-end security, end-nodes have an implementation of a symmetric key algorithm. For the low resource-devices, for example, battery-powered devices, the cryptographic operation with a limited amount of energy consumption is important. Application of the lightweight symmetric key algorithm allows lower energy consumption for end devices.

4.5.2 Applicability to Lower Resource Devices

The footprint of the lightweight cryptographic primitives is smaller than the conventional cryptographic ones. The lightweight cryptographic primitives would open possibilities of more network connections with lower resource devices.

A comparison of the lightweight properties with the conventional cryptographic primitives is shown in Table 4.3. The comparison in Appendix focuses on hardware properties. Some end-nodes might be able to embed general-purpose microprocessors and software properties are considered important in such platforms. However, lowest cost devices can embed only application-specific ICs due to limited cost and power consumption, where hardware properties are crucially important.

Cryptographic technologies are advancing: new techniques on attack, design, and implementation are extensively studied. One of the state-of-the-art techniques is "Lightweight Cryptography (LWC)." Lightweight cryptography is a cryptographic algorithm or protocol tailored for implementation in constrained environments including RFID tags, sensors, contactless smart cards, healthcare devices, and so on.

The properties of lightweight cryptography have already been discussed in ISO/IEC 29192 in ISO/IEC JTC 1/SC 27. ISO/IEC 29192 is a new standardization project of lightweight cryptography, and the project is in process of standardization. In ISO/IEC 29192, lightweight properties are described based on target platforms. In hardware implementations, chip size and/or energy consumption are the important measures to evaluate the lightweight properties. In software implementations, the smaller code and/or RAM size are preferable for the lightweight applications. From the view of the implementation properties, the lightweight primitives are superior to conventional cryptographic ones, which are currently used in the Internet security protocols, for example, IPsec, TLS.

Lightweight cryptography also delivers adequate security. Lightweight cryptography does not always exploit the security-efficiency trade-offs. We report recent technologies of lightweight cryptographic primitives.

Table 4.3 Results on Hardware Performance

	Mode	Block Size (Bits)	Key Size (Bits)	Cycle	Area (GE)	Frequency (MHz)	Throughput (Mbps)	Technology (μm)
Serialized Implementation (Area Optimization)								
PRESENT	enc	64	80	547	1075	0.1	0.0117	0.18
PRESENT	enc	64	128	559	1391	0.1	0.0115	0.18
CLEFIA	enc	128	128	176	2893	67	49	0.13
CLEFIA	enc/dec	128	128	176	2996	61	44	0.13
AES	enc	128	128	177	3100	152	110	0.13
AES	enc/dec	128	128	1032	3400	80	10	0.35
Round-Based Implementation (Efficiency Optimization)								
PRESENT	enc	64	80	32	1570	0.1	0.20	0.18
PRESENT	enc	64	128	32	1884	0.1	0.20	0.18
CLEFIA	enc/dec	128	128	36	4950	201.3	715.69	0.09
CLEFIA	enc/dec	128	128	18	5979	225.8	1605.94	0.09
AES	enc/dec	128	128	11	12,454	145.4	1691.35	0.13
AES	enc/dec	128	128	54	5398	131.2	311.09	0.13

Lightweight cryptography contributes to the security of smart objects networks because of its efficiency and smaller footprint. We believe that lightweight primitives should be considered to be implemented in the networks. Especially, lightweight block ciphers are practical to use now (Table 4.3).

4.6 EXISTING SECURITY SCHEMES FOR IoT

In existing networks, a number of data protection solutions have been applied for protection of data. In IoT environment, security still is a big concern. In IoT, from the nodes to the applications, the security challenges have posed. Fig. 4.10 shows a brief architecture of an IoT systems.

The typical security scheme should be addressed throughout the node life cycle from the initial design to the operational environment.

Secure boot: It is a process involving cryptography that allows an electronic device to start executing authenticated and trusted software to operate. To implement a secure boot with the help of public-key-based signature verification, a basic procedure is as follows. It is the foundation of trust but the nodes still need protection from various run-time threats and malicious intentions.

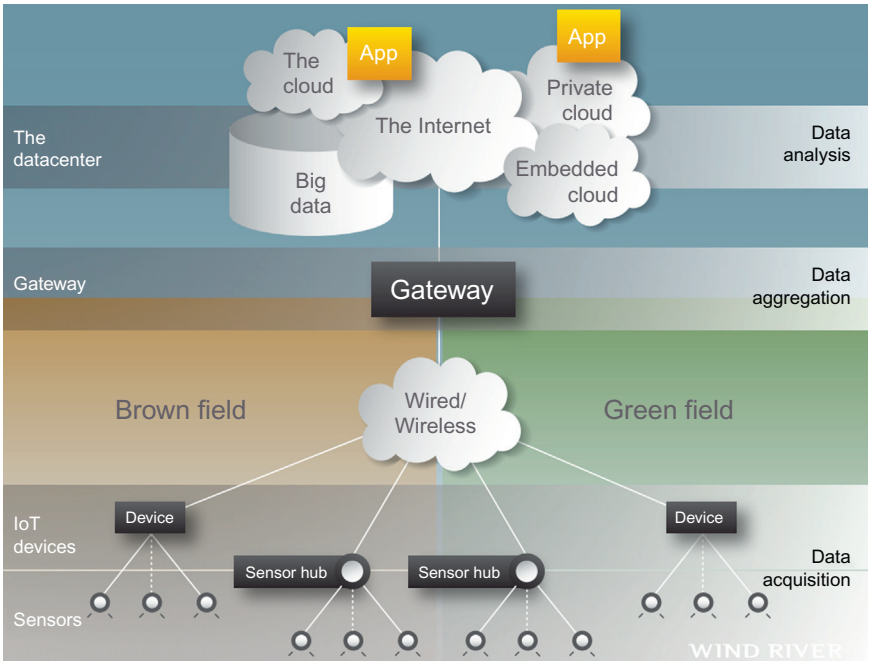

FIGURE 4.10
Structure of an IoT system.

Access control: The access control should be well designed to mandatory different forms of resources and roles in IoT. Basically, the privilege dictates that only the minimal access required to perform a function should be authorized in order to minimize the effectiveness of any breach of security.

Existing PKC schemes verify the integrity and authenticity of digital contents. As mentioned above, the integrity means that the digital content has not be been modified since it was created. Authenticity means that the same digital content has been released by a well-identified entity. The digital signature provides the two fundamental characteristics to make sure the digital content is trusted by other entity.

1. Integrity of digital content is guaranteed by message digest, that is, a secure hash algorithm (SHA-1, SHA-256, SHA-3, etc.).
2. The authenticity of digital content is guaranteed by the public-key-based signature scheme itself. PKC is based on pairs of keys. Anyone can possess a pair of keys: one private key stored secretly (K_PRIV), and one public key (K_PUB) publicly available to anyone. The K_PRIV can be used to sign digital content. The issuer of the digital content uses its own K_PRIV to identify himself/herself as the "issuer," the public key can be used by anyone to verify a digital content's signature.

a. Hash. Hashing the digital content and producing a hash value with the properties.
b. Sign. The hash value is signed (encrypt hash using K_PRIV) using the K_PRIV of the digital content author. The procedure value is called "signature" that is attached to the original digital content.

   Verify. If one wants to verify the digital content signature they have to perform following two steps:
c. Hash again. The digital content is hashed again, as in the signature generation process.
d. Reconstructed hash value is used as an input to the signature verification algorithm together with the signature attached to the digital content and the K_PUB (decrypt using signer's K_PUB) (Fig. 4.11).

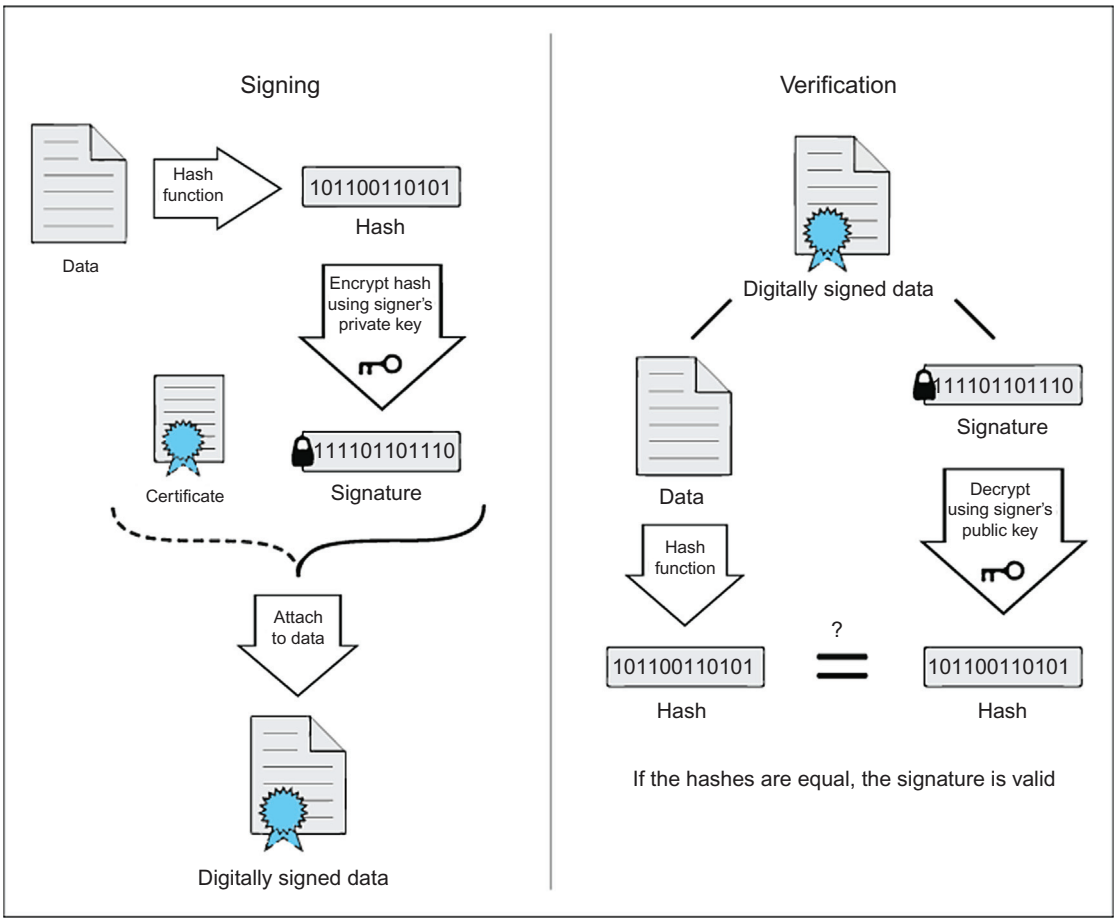

FIGURE 4.11
Example of digital sign system.

4.7 SUMMARY

The IoT is growing quickly and a number of smart objectives are bringing together, which can bring vulnerabilities in to the IoT systems and may carry serious risks for IoT devices, users, and for IoT-based applications. The hardware-based security solution can secure IoT systems and prevent damages and economic losses offering new opportunities. The IoT hardware security architecture is still in its exploratory stage, so it is facing many severe challenges than expected.

Further Reading

Fleisch, E., 2010. What is the internet of things? An economic perspective. Econom., Manage. Financ. Markets (2), 125–157.

Floerkemeier, C., Roduner, C., Lampe, M., 2007. Rfid application development with the accada middleware platform. IEEE Syst. J. 1 (2), 82–94.

Furnell, S., 2007. Making security usable: are things improving? Comput. Security 26 (6), 434–443.

Gama, K., Touseau, L., Donsez, D., 2012. Combining heterogeneous service technologies for building an internet of things middleware. Comput. Commun. 35 (4), 405–417.

Gaur, H., 2013. Internet of things: thinking services.

Gu, L., Wang, J., Sun, B., 2014. Trust management mechanism for internet of things. China Commun. 11 (2), 148–156.

He, W., Xu, L., 2012. Integration of distributed enterprise applications: a survey.

Hendricks, K.B., Singhal, V.R., Stratman, J.K., 2007. The impact of enterprise systems on corporate performance: A study of erp, scm, and crm system implementations. J. Operat. Manage. 25 (1), 65–82.

Hepp, M., Siorpaes, K., Bachlechner, D., 2007. Harvesting wiki consensus: Using wikipedia entries as vocabulary for knowledge management. IEEE Internet Comput. 11 (5), 54–65.

Hernandez-Castro, J.C., Tapiador, J.M.E., Peris-Lopez, P., Li, T., Quisquater, J.-J., 2013. Cryptanalysis of the sasi ultra-light weight rfid authentication protocol. arxiv.

Hitt, L.M., Wu, D., Zhou, X., 2002. Investment in enterprise resource planning: Business impact and productivity measures. J. Manage. Informat. Syst. 19 (1), 71–98.

Hoyland, C.A., Adams, K.M., Tolk, A., Xu, L.D., 2014. The rq-tech methodology: a new paradigm for conceptualizing strategic enterprise architectures. J. Manage. Analyt. 1 (1), 55–77.

HP Company, 2014. Internet of things research study. Retrieved from <http://digitalstrategies.tuck.dartmouth.edu/cds-uploads/people/pdf/Xu_IoTSecurity.pdf>.

ITU, 2013. The internet of things, international telecommunication union (itu) internet report.

Kang, K., Pang, Z., Da Xu, L., Ma, L., Wang, C., 2014. An interactive trust model for application market of the internet of things. IEEE Transact. Indust. Informat. 10 (2), 1516–1526.

Keoh, S., Kumar, S., Tschofenig, H., 2014. Securing the internet of things: a standardization perspective.

Kim, H., 2012. Security and vulnerability of SCADA systems over ip-based wireless sensor networks. International J. Distribut. Sensor Networks 8 (11), 268478.

Klair, D.K., Chin, K.-W., Raad, R., 2010. A survey and tutorial of rfid anti-collision protocols. IEEE Commun. Surv. Tutor. 12 (3), 400–421.

Kranenburg, R.v., Anzelmo, E., Bassi, A., Caprio, D., Dodson, S., Ratto, M., 2011. The internet of things. Paper presented at the 1st Berlin Symposium on Internet and Society (Versión electrónica). Consultado el.

Li, D.X., 2011. Enterprise systems: state-of-the-art and future trends. IEEE Transact. Indust. Informat. 7 (4), 630–640.

Li, F., Xiong, P., 2013. Practical secure communication for integrating wireless sensor networks into the internet of things.

Li, L., Li, S., Zhao, S., 2014. Qos-aware scheduling of services-oriented internet of things.

Li, L., Wang, B., Wang, A., 2014. An emergency resource allocation model for maritime chemical spill accidents. J. Manage. Analyt. 1, 146–155.

Li, S., Da Xu, L., Zhao, S., 2014. The internet of things: a survey. Informat. Syst. Front. 17, 243–259.

Lim, M.K., Bahr, W., Leung, S.C., 2013. Rfid in the warehouse: a literature analysis (1995–2010) of its applications, benefits, challenges and future trends. Int. J. Product. Econom. 145 (1), 409–430.

Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I., 2012. Internet of things: vision, applications and research challenges. Ad Hoc Networks 10 (7), 1497–1516.

Ning, H., 2013. Unit and Ubiquitous Internet of Things. CRC Press, Boca Raton, FL.

Ning, H., Liu, H., Yang, L.T., 2013. Cyberentity security in the internet of things. Computer 46 (4), 46–53.

Oppliger, R., 2011. Security and privacy in an online world. Computer 44 (9), 21–22.

Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A., 2006. M2ap: a minimalist mutual-authentication protocol for low-cost rfid tags. Ubiquitous Intelligence and Computing. Springer, Heidelberg, pp. 912–923.

Perna, M., 2013. Security 101: securing SCADA environments. Retrieved from <http://blog.fortinet.com/post/security-101-securing-scada-environments>.

Pretz, K., 2013. The next evolution of the internet. Retrieved from <http://theinstitute.ieee.org/technology-focus/technology-topic/the-next-evolution-of-the-internet>.

Raza, S., Voigt, T., Jutvik, V., 2012. Lightweight ikev2: a key management solution for both the compressed ipsec and the ieee 802.15. 4 security. Paper presented at the Proceedings of the IETF Workshop on Smart Object Security.

Raza, S., Shafagh, H., Hewage, K., Hummen, R., Voigt, T., 2013. Lithe: lightweight secure coap for the internet of things.

Roe, D., 2014. Top 5 internet of things security concerns. Retrieved from <http://www.cmswire.com/cms/internet-of-things/top-5-internet-of-things-security-concerns-026043.php>.

Roman, R., Najera, P., Lopez, J., 2011. Securing the internet of things. Computer 44 (9), 51–58.

Roman, R., Zhou, J., Lopez, J., 2013. On the features and challenges of security and privacy in distributed internet of things. Comput. Networks 57 (10), 2266–2279.

Sundmaeker, H., Guillemin, P., Friess, P., Woelfflé, S., 2010. Vision and challenges for realising the internet of things: EUR-OP.

Tan, W., Chen, S., Li, J., Li, L., Wang, T., Hu, X., 2014. A trust evaluation model for e-learning systems. Syst. Res. Behav. Sci. 31 (3), 353–365.

Tao, F., Cheng, Y., Xu, L.D., Zhang, L., Li, B.H., 2014. Cciot-cmfg: cloud computing and internet of things based cloud manufacturing service system.

Wang, F., Ge, B., Zhang, L., Chen, Y., Xin, Y., Li, X., 2013. A system framework of security management in enterprise systems. Syst. Res. Behav. Sci. 30 (3), 287–299.

Wang, K., Wu, M., 2010. Cooperative communications based on trust model for mobile ad hoc networks. IET Informat. Security 4 (2), 68–79.

Weber, R.H., 2013. Internet of things–governance quo vadis? Comput. Law Security Rev. 29 (4), 341–347.

Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., et al., 2009. Building the internet of things using rfid: the rfid ecosystem experience. IEEE Internet Comput. 13 (3), 48–55.

Wieder, B., Booth, P., Matolcsy, Z.P., Ossimitz, M.-L., 2006. The impact of erp systems on firm and business process performance. J. Enterprise Informat. Manage. 19 (1), 13–29.

Xiao, G., Guo, J., Xu, L., Gong, Z., 2014. User interoperability with heterogeneous iot devices through transformation.

Xu, B., Xu, L.D., Cai, H., Xie, C., Hu, J., Bu, F., 2014. Ubiquitous data accessing method in iot--based information system for emergency medical services.

Xu, L., He, W., Li, S., 2014. Internet of things in industries: a survey. IEEE Transact. Indust. Informat. 99, 1.

Xu, L.D., 2011. Information architecture for supply chain quality management. Int. J. Product. Res. 49 (1), 183–198.

Yao, X., Han, X., Du, X., Zhou, X., 2013. A lightweight multicast authentication mechanism for small scale iot applications.

Yuan Jie, F., Yue Hong, Y., Li Da, X., Yan, Z., Fan, W., 2014. Iot-based smart rehabilitation system. IEEE Transact. Indust. Informat. 10 (2), 1568–1577.