Trust Your Digital Certificates—Even When Offline

Introduction

Our world is getting more connected every day. The Internet of Things (IoT) revolution will only succeed if users can trust connected devices and their underlying infrastructure. In the past, security was a concern only for dedicated applications, such as electronic payment systems. Today, however, security has become a requirement for many additional applications, such as smart grid, process control, and building automation (Figure 1).

One of the fundamental security requirements for a connected device is the ability to authenticate the source of received information. This can be achieved using digital certificates. There are challenges with maintaining digital certificates for connected devices in the field. In this design solution, we will review the online operation of digital certificates and show how they can now be kept secure and up-to-date throughout the lifecycle of connected devices, even when the devices are offline.

Figure 1. Remote Smart Home Control

Online Authentication

Strong authentication depends on the use of a cryptographic algorithm and a key. There are two basic types of cryptographic algorithms: symmetric and asymmetric. A symmetric algorithm, such as the HMAC, requires the sender and receiver to have a shared secret. The secret is often referred to as a key, which must be stored securely and must never be disclosed to a third party. Since both the sender and the receiver authenticate data using the same secret key, they can exchange information with the knowledge that the information source is authentic. The drawback of the symmetric scheme is that the keys might be exposed when they are distributed, thus creating a major vulnerability.

An alternative is to use an asymmetric (also known as public key) algorithm, such as RSA or ECDSA. Asymmetric algorithms use two keys—one is stored privately and the other is made public for anyone to use—making the distribution of keys much easier. (Note that data signed with a private key can only be verified with its associated public key.)

An example of sending a message using Asymmetric Cryptography is shown in Figure 2. If Alice wishes to send trusted information to Bob, she signs it with her private key. However, unlike with the symmetric scheme, Bob does not need Alice's secret private key to verify that the message was truly sent by Alice. He can verify the message using Alice's public key. As its name suggests, a public key is not a secret and can be freely disclosed, and easily distributed to whoever wishes to use it. Once Bob has verified the signature of the message, he can be sure it is from Alice since only Alice's public key can be used to perform the verification.

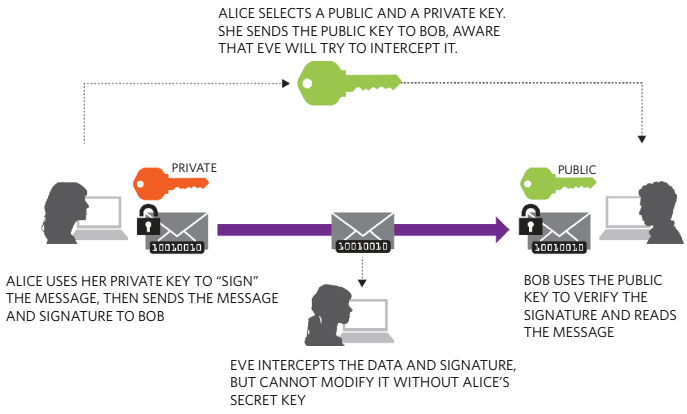

Figure 2. Sending a Message Using Asymmetric Cryptography

But there is still a risk that an attacker could replace Alice's public key with his or her own key. In such a scenario (Figure 3), Eve, masquerading as Alice, sends her public key to Bob pretending it is Alice's. She further sends messages to Bob signed with her key. Because Eve's public key verifies messages signed using her own private key, Bob would mistakenly believe he is communicating with Alice. This is possible because Bob has no way to verify the identity of anyone who sends him public keys.

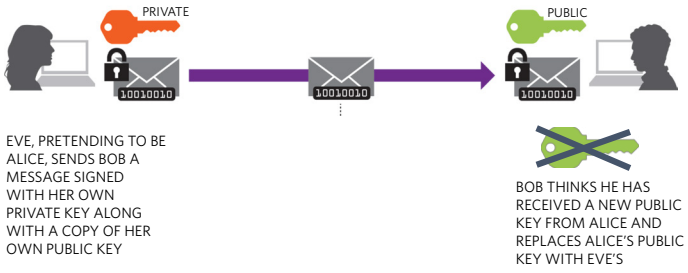

Figure 3. Intruder Substituting a Public Key

Using public key infrastructure (PKI) prevents this kind of attack. With PKI, the validity of a public key is guaranteed by a trusted third-party Certification Authority (CA). A CA signs and issues a digital certificate for third-party public keys. A digital certificate is a file containing a public key and ID information that can be used to verify the ownership of the key. This certificate is then signed by a CA using their own private key.

For example, in Figure 4, Alice attaches ID information to her public key, and the trusted CA signs the certificate using their private key, thereby creating Alice's trusted digital certificate. She sends a copy of this digital certificate to Bob. He uses the public key of the CA to verify that the certificate can be trusted (since the certificate was signed by the private key of the trusted CA). Bob then add Alice's certificate to his list of trusted certificates. He can then use Alice's public key to verify that future messages he receives from Alice truly came from her. If Bob receives a certificate attached that cannot be verified by the CA, then that certificate is discarded and any further messages received with this attached certificate are ignored.

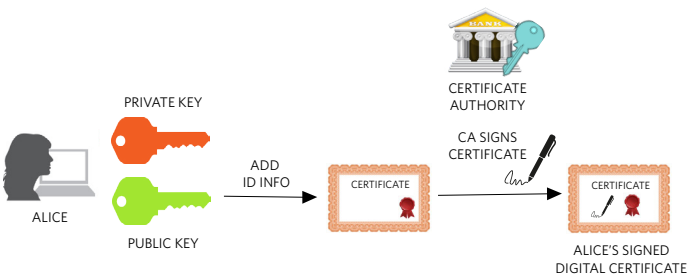

Figure 4. Using a CA to Generate a Digital Certificate

It should be noted that the validity of a CA might be guaranteed, in turn, by the public key of a higher-level CA. In this case, at the top of this hierarchy is a "root key" or a securely stored public key, which cannot be modified or interfered with by an unauthorized person.

PKIs are widely used for website authentication. Home banking, for example, relies heavily on a PKI and the connection to a CA. Typically, the green lock next to https:// in an internet browser address means that the authenticity of the webpage has been verified through a certificate. This online verification process is largely transparent to the website user.

Offline Authentication

A connected device is one that communicates with a remote server across a network or the internet. A simple example of this (Figure 5) might be a network-connected programmable logic controller (PLC) in an industrial setting. In such a setting, the network is usually private and may not be directly connected to the internet and may not have a connection to a trusted third-party CA. Even if the network is connected to the internet, the network administrator may choose not to use the services of a third-party CA, due to costs or other reasons.

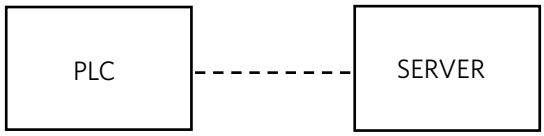

Figure 5. A Network-Connected PLC

Occasionally, it may be necessary to upload new software or configuration information to the PLC. Even if the network is private, it is still important for the PLC to be able to verify the authenticity of the data source. Otherwise, an intruder who gains access to the network could attempt to download software or fake configuration data to the PLC. To prevent this, digital certificates can be used to verify the authenticity of information sent to the PLC. However, if there are no means to verify the certificates (for example, if there is no internet connection), an intruder could upload a false digital certificate to the list of trusted certificates on the PLC and begin communicating with it.

To prevent this from happening, it is necessary to provide a means of verification within the PLC itself. Clearly, it is not a trivial task to establish a secure form of self-certification. The reality is that many PLCs (and other connected devices) do not have the necessary security features to verify the authenticity of any digital certificates. This presents the potential for a significant security breach within the network.

Secure Microcontroller

A simple solution to this problem is to use an additional secure microcontroller which incorporates authentication. The MAXQ1061 DeepCover® cryptographic controller allows a connected device manufacturer to establish its own CA for all its connected devices (for example, for future firmware updates). The device manufacturer generates a private/public key pair, then securely stores the root public key in the EEPROM of the MAXQ1061. As a service, the public key can also be programmed by Maxim on behalf of the customer. The hardware security features and the secure file system of the MAXQ1061 prevent unauthorized users from modifying the key. This root key sits at the top of the CA hierarchy of the device manufacturer for all future "child" certificates received by the device.

Figure 6 illustrates the steps for the MAXQ1061 to verify a newly issued child certificate. First, the new child certificate is generated containing the public key and the ID of the user wishing to communicate with the connected device. Second, the child certificate is signed offline by the connected device manufacturer, using its system private key to create a digital certificate for the user. Next, upon receiving this certificate, the MAXQ1061 verifies the signature using the stored root public key. Finally, if verification was successful, the MAXQ1061 stores it as a "certificate object" in EEPROM. The verified certificate is trusted for all future communications from this user until it is revoked. In this way, new child certificates can be securely loaded at any time, and existing certificates can be easily updated or revoked. Hence, the device manufacturer effectively acts as an offline CA for trusted users of its devices in the field.

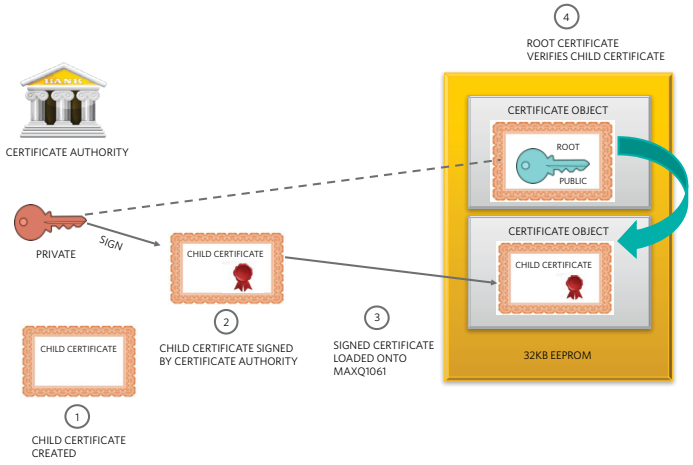

Figure 6. Verifying a Certificate with the MAXQ1061

Unique Benefits of the MAXQ1061

The MAXQ1061 allows several certificate trees and CAs to be employed simultaneously. Referring to the previous PLC example, it may be necessary for the PLC to authenticate information from two different sources such as:

- Authentication of the PLC manufacturer for operating system upgrades
- Authentication of the PLC programmer for program or user data updates

In this scenario, separate root keys would be required, one for the PLC manufacturer and one for the programmer. The MAXQ1061 manages the root keys in such a way that independent child certificates do not interfere with each other. Not only does the MAXQ1061 make the distribution and update of the certificates possible, it also has the necessary computing power to perform the resource-intensive signature verification operations. This is especially useful when the main microcontroller of the connected device does not have the cryptographic functionality or the computing power to handle these computationally intensive functions.

Another notable benefit of the MAXQ1061 is the ability to define and store different types of objects in the on-board 32KB EEPROM. Examples include:

- Its own certificates that have been signed by a CA and the associated private keys. These can be used to provide authentication to peer devices. Each certificate is protected against modification, and each private key against disclosure.
- Root CA public keys that can be used to verify peer certificate chains since the CA public keys cannot be modified.
- Arbitrary symmetric keys, public keys and private keys used in cryptographic algorithms.
- Administrator authentication keys and the host authentication keys (for an optional secure channel).
- Any additional arbitrary data up to the limit of the storage size.

Objects are stored in hardware-protected nonvolatile memory and object modification is atomic, so that if a write operation is interrupted (for example due to a power loss), the object will retain its previous known value. Objects can be allocated and de-allocated as required and the free space reclaimed. A stored object has separate access conditions (read, write, execute, etc.) depending on the lifecycle state of the device (initialized, operational, etc.)

Conclusion

In this Design Solution, we reviewed the requirements for secure authentication through certificates by connected devices. We highlighted the vulnerability of attack for non-CA-supported devices, which also have no way to verify the authenticity of the source of incoming communications.

The MAXQ1061 DeepCover cryptographic controller was presented as a simple solution to this problem. In addition to certificate distribution and management, the unique benefits of the MAXQ1061 can be applied to secure access control (such as for home automation), electronic signature generation, and cybersecurity for critical infrastructures, including the smart grid. The MAXQ1061 is also an ideal solution for manufacturers of IoT devices who do not want to rely on the services of a third-party CA but instead wish to implement self-certification of digital certificates.

Glossary

IoT: Internet of Things. Remote microcontroller-driven devices with access to the internet for configuration or control.

CA: Certification authority. Independent trusted organizations responsible for verifying the authenticity of public keys**.**

PKI: Public key infrastructure. A set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption**.**

PLC: Programmable logic controller. A computer dedicated to the control of industrial processes.

SHA: Secure hash algorithm

RSA: Rivest shamir adleman. A public key encryption algorithm.

ECDSA: Elliptic curve digital signature algorithm. A public key encryption algorithm.

Learn more:

MAXQ1061 DeepCover Cryptographic Controller for Embedded Devices

Design Solutions No. 56

Rev 0; May 2017

Need Design Support?
Call 888 MAXIM-IC (888 629-4642)

Find More Design Solutions

Maxim Integrated
160 Rio Robles
San Jose, CA 95134 USA
408-601-1000

maximintegrated.com/design-solutions