Packing Up Cryptography to Protect IoT

By Mark Partick, Mouser Electronics

It's no secret that the Internet of Things (IoT) will impact almost every facet of the way people live, work, travel and access services. Businesses, for their part will rely heavily on information acquired through the IoT to support customers, improve processes, drive cost savings and inspire new product development.

Whether, ultimately, there will be 10, 50 or 100 billion 'things' connected to the Internet, they will constitute a huge number of remotely accessible devices that contain valuable information and intellectual property in the form of software code. They also have access privileges to information-laden servers and corporate networks. Furthermore, there are already plenty of them out there, each being potentially exposed to unauthorised attempts to take advantage of any related assets.

Protecting these devices against attack, and being able to detect and isolate any that are found to be compromised, is critical if the world is to realise the tremendous advantages the IoT has to offer. Using asymmetric cryptography to protect device boot-up and authentication, and keep data and intellectual property secret, can provide the lightweight and robust security that the IoT needs.

Protection from Power-Up to End of Life

An IoT device is vulnerable from the moment it is installed and powered up. If hackers can interfere with the boot process, they can insert malware for numerous purposes, such as snooping on legitimate users, stealing the intellectual property associated with the application code, or taking over the device entirely. Having gained control of a device, they can falsify information, disable equipment, sabotage processes or safety systems, or try to access corporate networks to steal sensitive information.

A conventional boot sequence verifies only that the application image is present, and performs a checksum or CRC check before loading the application. This process would be open to an attempt to replace the application image with malware. A secure boot process, in contrast, first verifies the authenticity of the application image. If the image is not authenticated, the boot sequence is not permitted to start. Verifying the authenticity of the code means the device can be trusted to behave as expected, and perform only known and legitimate functions. In this way, securing the boot process establishes a root of trust in the device.

The root of trust provides a basis for the device to authenticate itself when connecting to the network. Security depends on mutual authentication, which means the device must also be able to authenticate its host. Establishing mutual trust between the device and the network creates the foundation for secure interactions. The methods used to do this rely on various forms of symmetric and asymmetric cryptography, such as the Elliptic Curve Diffie Hellman (ECDH) protocol for encryption/decryption, Elliptic Curve Digital Signature Algorithm (ECDSA) for sign/verify authentication, and symmetric challenge/response leveraging a hash algorithm, like SHA-256, to prevent key derivation.

To have a complete strong authentication from end-to-end, an IoT secure platform must also include a managed in-manufacturing provisioning for ensuring assigned identities are not corrupted during the development process.

Crypto-Centric Security: Benefits and Barriers

Cryptographic algorithms are already widely used to protect conventional Internet transactions, such as online payments, and provide a way of verifying the source of data and preventing the information contained from being intercepted. One example is the use of asymmetric cryptography, based on a combination of both private and public keys, to setup an encrypted communication session between a web browser and server. When a browser wants to start a session, the server sends a copy of its own digital certificate issued by a trusted certificating authority. The certificate authenticates the server and presents the server's public encryption key, which the browser uses to create a session key. The server then decrypts the key using its own private key, and the encrypted session can begin.

As far as IoT use cases are concerned, however, the use of cryptographic techniques, either symmetric or asymmetric, presents a number of challenges. One is that the cryptographic algorithms used to encrypt/decrypt transmissions are inherently computationally intensive - if not beyond the capabilities of the average IoT-device MCU, it will certainly be an unwanted extra load on system resources that are by the very nature of IoT extremely limited. Also, the Flash or ROM memories of the standard MCU that are likely to be used in basic IoT devices are not a secure place for storing public or private cryptographic keys.

In addition to all this, exchanging encrypted messages imposes transmission overheads that are significantly greater than the data payload of typical IoT devices, such as smart sensors - which will normally only send a few bytes of data at a time, at infrequent intervals. Moreover, a robust Public Key Infrastructure (PKI) is needed, which is capable of keeping private keys private and ensuring the authenticity of digital certificates. A small number of publicly recognized authorities provide PKI services to keep conventional Internet transactions, such as online payments, secure. However, the same model is not suitable for closed IoT ecosystems. An economical and convenient means of generating and protecting keys and digital certificates, plus maintaining records, is going to be mandated.

In Search of an IoT-Friendly Solution

An embedded secure element connected to the main microcontroller, containing secure data storage and encryption/decryption algorithms implemented in hardware, can provide robust protection within the typical system constraints of a small IoT device. The host CPU interacts with the secure element via APIs to command security services as needed without exposing any secrets. The secure element integrates all the necessary protection against physical attacks and side-channel attacks. Hence the microcontroller can be a general-purpose one, chosen to fulfil the functional requirements of the IoT device, and does not need to be an expensive secure microcontroller with high-compute performance to execute cryptographic algorithms in software.

An example of this type of IC is the Microchip ATECC608A - which provides services to support authentication, integrity, and confidentiality in resource-constrained connected devices. It features an integrated cryptographic hardware accelerator that can perform asymmetric cryptography operations up to 1000x faster than software running on standard microprocessors. At the same time, the ever-present risk of key exposure when using a standard microcontroller is greatly reduced.

The IC handles identity authentication and session key creation, and supports multiple key-generation protocols - including TLS 1.2 and TLS 1.3. There is also support for small-message encryption using symmetric AES-ECB Block Cipher mode, which helps minimize communication overheads when encrypting or decrypting small messages or personally identifiable information. Key generation for software download is another valuable built-in feature for IoT device management, to authenticate firmware updates distributed on a broadcast basis or point-to-point across the network.

Figure 1: The ATECC608A from Microchip

Secure Boot and Certificate Authority Services

By supporting ECDSA signing, the ATECC608A can secure the boot sequence of IoT devices, and so help establish the root of trust upon power-up. The encrypted application image, and its digest - digitally signed using ECDSA with a private key, are loaded on the device microcontroller. The secure element stores a public key derived from the private signing key, and can operate in various modes to verify the digest, or the signature, or both, using the public key. The boot sequence is allowed to start upon successful verification, or is prevented in the event of failure.

The ATECC608A instruction set simplifies exercising the key generation, encryption and secure-boot functionality, and the IC comes with Microchip's CryptoAuthentication™ development library containing a set of APIs for core interactions, handling certificates and TLS integration.

To complete the IoT-device secured solution, a complete in-manufacturing provisioning service is provided by Microchip. This includes public/private key generation with secure storage and record-keeping, which saves device manufacturers the considerable burden of setting up and maintaining their own secure facilities. The ICs are shipped pre-provisioned from secure facilities, ready for final assembly, which eliminates any need to share secrets with software or manufacturing third-parties.

Conclusion

The IoT can offer tremendous benefits for individuals and businesses. However, its pervasive, connected nature, and reliance on autonomous, resource-restricted devices, present unique security challenges. Proven cryptographic techniques are needed, but must be implemented within the tight confines in terms compute power, price point, and energy consumption. Offloading these functions to a dedicated embedded secure element enables a practicable, cost-effective and robust solution to be realised.