Whitepaper

Seal SQ
Secure IoT Device to Cloud solution

Contents

About SEAL SQ

Seal SQ is a pure play cybersecurity company, with over 20 years of experience in providing digital trust and cryptographic protection. Seal SQ delivers secure semiconductors, digital certificates, digital IDs, as well as SaaS platforms for proof-of-provenance, lifecycle management and blockchain-driven traceability. Seal SQ customers are typically IoT vendors servicing smart buildings, smart cities, smart agriculture, drones, health care monitoring, logistics and Industry 4.0. Seal SQ has even been able to successfully extend its Trust model to non-connected objects. Such objects connect through NFC or a plug when needed, and include luxury goods, health care consumables, appliance accessories, cold crypto wallets, and pieces of art.

Seal SQ's certificate Authorities, Security Brokers, management systems and tamper resistant secure microcontrollers are regularly audited and accredited with highest grade WebTrust, Common Criteria and FIPS certifications.

Introduction

In 2018, the global market for the Internet of Things (IoT) reached $130bn. It is projected to reach $318bn by 2023 at a compound annual growth rate (CAGR) of 20%. According to IHS Markit, more than 75 billion IoT devices will be online by 2025. Yet, for all its promise, IoT technology is not without its difficulties and challenges. Early adopters of IoT technologies encountered significant barriers to adoption.

Security tops the list of major concerns, holding back 59% of those professionals from proceeding. According to a study by the Ponemon Institute, 63% of CISO's believe that participation in IoT will increase cybersecurity risks in the future, and that over 80% of professionals predict that their organization will experience a catastrophic data breach caused by an unsecured IoT device. In this paper we address a complex fundamental component of securing IoT: how to give devices and services secure identities so they can interact securely.

Threats and vulnerabilities of IoT

Security for the IoT is increasingly being recognized as a critical requirement; not only due to the logical and physical security aspects that are so closely entwined with digital security, but also because of the need to effectively manage deployments. This requires identification for the purposes of authentication, attestation, and access control, at a minimum.

For most IoT deployments, a trusted ecosystem of authorized devices and authorized services is the recommended approach. In a trusted ecosystem unauthorized devices or services are not allowed to interact with authorized devices or services. This prevents illicit access to the critical services and data of an IoT device.

Establishing a trusted ecosystem is not possible without mutual authentication. This allows two entities to prove to each other that they are authorized members of a particular trusted ecosystem. The standard practice for mutual authentication is public key cryptography.

Seal SQ secure device-to-cloud solution

Key components

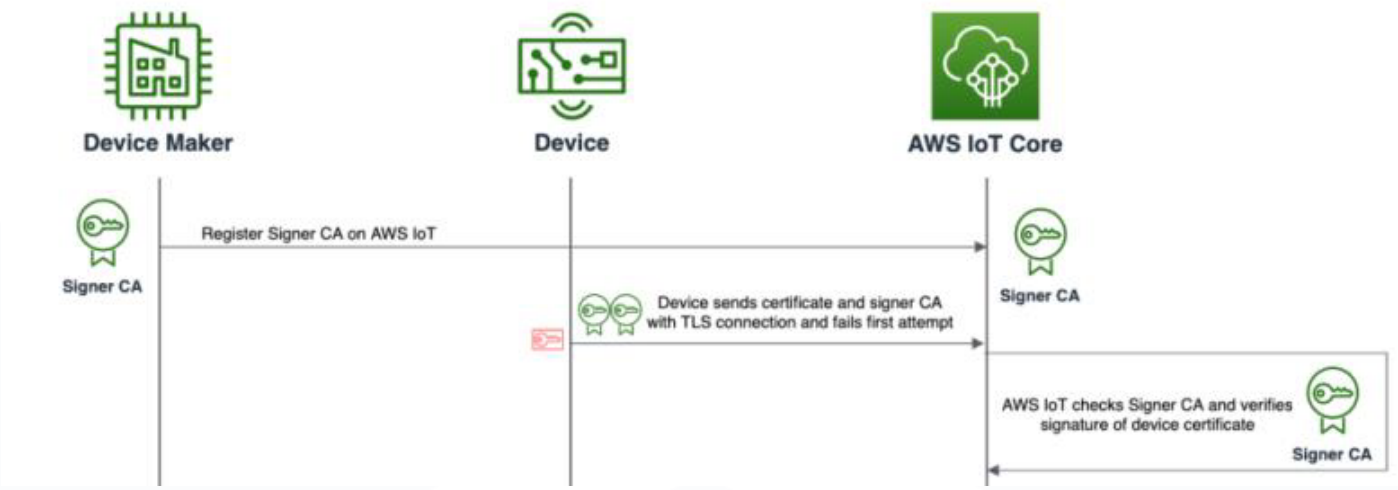
Public Key Cryptography

A Public Key Infrastructure (PKI) is commonly defined as “a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates”, but the understanding of this definition requires us to explain some base concepts. PKI uses cryptographic techniques based on a pair of keys, with a unique mathematical relationship. In our case this relationship is based on Elliptic Curves: Elliptic Curve Cryptography or ECC. One key is private and kept secret, one key is public and can be disclosed to everybody. ECC works in such a way that a message signed (encrypted) with a private key can only be verified with the related public key. This technology is the foundation to build the four pillars of transaction security: confidentiality, authentication, integrity, and non-repudiation.

A subscriber or “end entity” (person, application or object) will have a key pair, a secret private key and a public key, usually presented in the form of a Digital Certificate.

In PKI the assurance on the identity of the owner of a private key is achieved by using an X.509 Digital Certificate that contains not only the public key but also the identity of the entity that has the corresponding private key, together with an attestation issued by an entity that verified the identity of the person or object according to a series of security practices. The entity that makes the attestation is known as the Certification Authority (CA) and the attestation itself has the form of a digital signature, generated with the private key of the CA. This implies one of the first concepts around trust: we can trust a Digital Certificate of an entity (person, application or object) only if we trust the CA that signed the Digital Certificate, or said differently, if we can verify the signature of the signing CA with its public key in which we trust.

It goes beyond the scope of this document to describe the various techniques to build trust in the signing CA, but one of them is to register the public key of the signing CA in a protected environment. This technique is often used in public cloud platforms such as AWS.



Root of Trust (Certificate Authority)

Another technique is to sign the certificate of the Signing CA with a publicly trusted Root of Trust. OISTE/Seal SQ's Swiss based cryptographic Root of Trust ("RoT") provides such publicly trusted secure authentication and identification, in both physical and virtual environments, for the Internet of Things, Brand Protection and NFT. The Seal SQ RoT serves as a common trust anchor to ensure the integrity of online transactions among objects and between objects and people.

Digital Identity

The essential component of a secure device-to-cloud-connection is the device's digital identity. This must be unique per device, hard to fake and hard to steal.

Once one can rely on these three characteristics of the digital identity, one can establish a complete trust chain for device authenticity and integrity, security of communication, and finally data trustworthiness.

What is needed to obtain a trusted digital identity of a device? First of all, a secure means to generate it, a secure means to use it in the device and, most and for all, a secure process to provision it in the device.

A digital identity consists of the combination of a unique serial number, or object identifier, linked to a secret private key and a digital certificate containing the related trusted public key.

A Certificate Authority is a well-protected environment where the link between object identifier and public key is asserted and certified by means of digital signature of this link. This is called a digital certificate.

The certification can be one-shot, a device receives a certificate that will be used for its authentication throughout its entire lifetime, or it can be dynamic in the sense that the certificate can be issued, revoked or renewed during the life of the device, depending on the needs of the network it is part of. For this purpose, a managed PKI system, or CMS (Certificate Management System) can be used, giving a system administrator the control over the lifecycle of the devices and their digital identities.

CMS (Certificate Management System)

The biggest barrier to deploying IoT projects is providing robust security that can be scaled for a large number of devices. The core security challenges are device personalization and lifecycle management. IoT devices need to be provisioned with unique, trusted identities that can interact in complex ecosystems of industrial and consumer IoT.

Seal SQ managed PKI (mPKI) service for IoT is called INeS CMS. It provides device personalization at the massive scale. Whether you want to deliver credentials on the factory floor or from an online cloud enabled service, Seal SQ makes scalable device personalization easy.

INeS CMS provides below key features:

- **Certificate Management** – It supports the definition of certificate templates, the generation of standalone certificates or batch certificates, and the management of the issued certificates (i.e. monitor, revoke, re-key).
- **CA management** – Users can configure the issuing CA for a specific organization.
- **Multi-tenancy** – It supports Role Based Access Control (RBAC) for the users of CMS and organization management.
- **Log auditing** – It logs each operation in CMS (i.e. user login, certificate request, certificate generation, etc).
- **Public cloud integration** – It integrates with public cloud services like AWS IoT Core and Azure DPS/IoT hub. IoT devices can easily on-board to a public cloud by using the certificates issued by INeS CMS for device attestation.
- **APIs support** – RESTful APIs and technical documents are available for automating the certificate enrolment process and managing the life-cycle of devices.
- **Client library support** – Client library and sample code are supported in different programming languages.

VaultIC Secure Elements

To store assets, like device private keys and associated digital certificates and possibly critical application specific data, Seal SQ recommends Secure Elements (SE), such as VaultIC408 or VaultIC292. These Secure Elements are designed specifically for the purpose of performing their pre-programmed security routines. These cryptographic services and functions, executed in a physically hardened environment, a tamper resistant chip, include low-level cryptographic methods and algorithms needed for authentication and data encryption/decryption as well as secure storage of essential data items.

The VaultIC family of products embeds cryptographic tool boxes for Authentication, Confidentiality and Integrity executed in a secure environment. The tool box proposes a variety of standard and NIST recommended algorithms and key lengths (e.g. ECC, RSA, ECDSA, AES, SHA...). VaultIC embeds on-chip tamper resistant data storage capabilities (NVM) for keys, certificates and customer data. VaultIC also features a True Random Number Generator, compliant to SP800-90B [NIST] to guarantee the entropy needed for high-level cryptographic services.

Seal SQ's VaultIC Secure Elements, provide multiple advantages when compared with software-based security, or security executed in specific trusted zones on the host MCU, including:

- Crypto keys and other security materials are stored under control of secure hardware, beyond the reach of any software attack on the host system, its RAM cache, or the VaultIC itself;
- Protection against physical attacks, also referred to as tamper-resistance, using various techniques like voltage, temperature or frequency sensors, bus encryption, shielding, memory access protection, countermeasures against power analysis, light detectors …. The VaultIC chips are able to protect their sensitive information up to a level that is certified by independent laboratories, according to Common Criteria and FIPS.
- Digital signature and verification like ECDSA are performed within the Secure Element.
- True Random Number Generation is performed within the Secure Element hardware, delivering true randomness that is vital for high quality generation of encryption keys – software alone is limited to Pseudo Random Number Generation, weakening encryption and security.

VaultIC408 is certified by the US FIPS 140-3 Level 3 standard [FIPS] and Common Criteria EAL 5+ [CC].

Building a secure connection to an IoT Cloud

Secure Provisioning of the Secure Element

The digital identity of a device, which is considered unique and authentic, should be generated and stored on a SE in a secure, controlled environment. This avoids the processing of this sensitive data in a device manufacturer's production chain and possible leakage of this sensitive information to a production of counterfeit devices. Because the digital ID is stored in the tamper resistant chip before leaving the chip manufacturer's secure production plant, the task of an OEM is merely to connect the chip on a board, amongst all other components. The simple fact that the secure element is tamper resistant is a guarantee that no secret data can leak during manufacturing and that the digital identity truly is unique.

This is depicted in the diagram below.

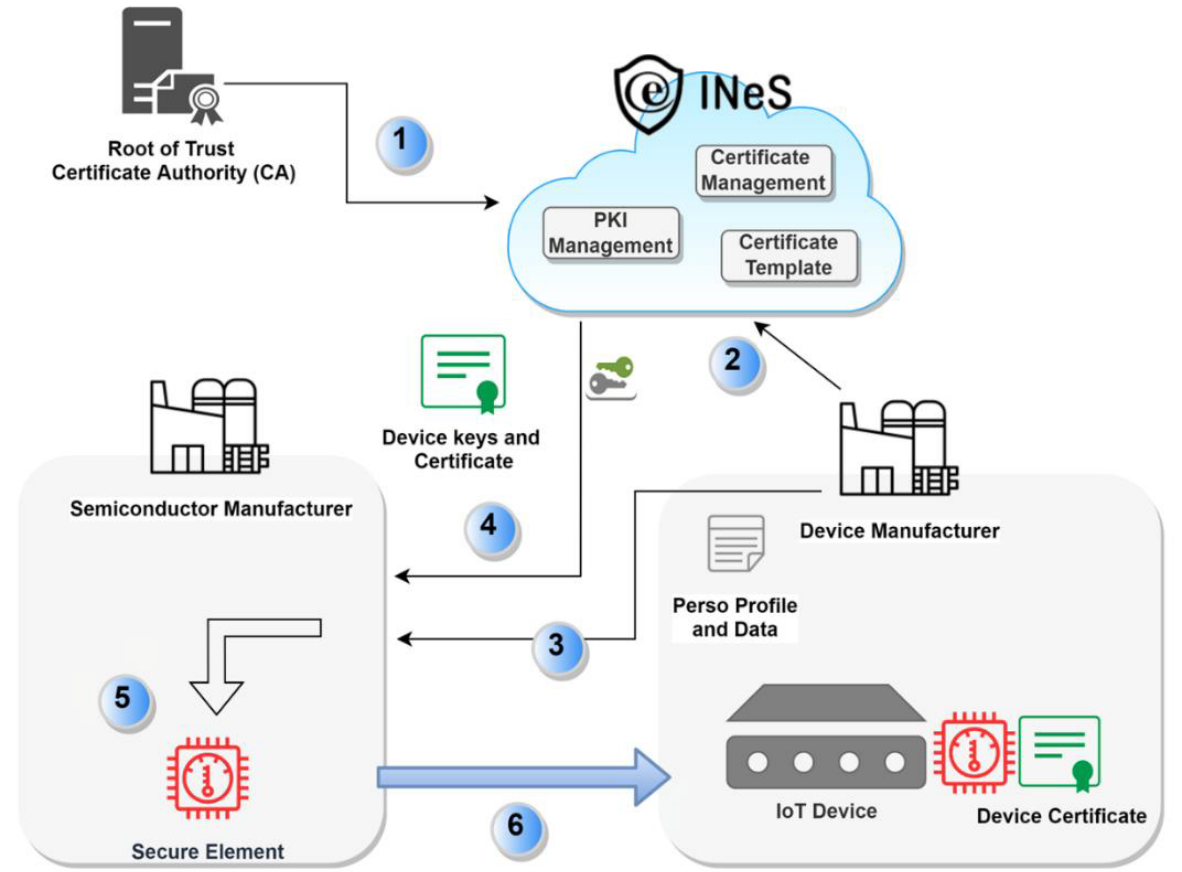


Figure 2: pre-provisioning of Digital Identity in Secure Element's production process

1. A Root Certificate Authority and an Issuing Certificate Authority (ICA) are setup by the Device Manufacturer

2. The device Manufacturer defines information such as device and certificate type and template for the devices to produce. The CMS can be asked to generate a batch of device private keys and certificates, signed by the ICA.

3. The device maker orders the Secure Element, and defines the memory profile and information that needs to be stored in the chip.

4. Seal SQ production retrieves from INeS the generated keys and certificates and

5. Stores them as required in the chips, in a Common Criteria certified secure production site.

6. At reception of the personalized chips, the device manufacturer or Contract Manufacturer only needs to build the physical device that will from the start possess an identity and can be authenticated for further operation.

Secure Cloud connection

In the IoT device, our SE (VaultIC408 in this case) is an accompanying chip that works along with the host MCU and stores the private key and device certificate for device authentication.

Below is the figure that illustrates the general workflow of how an IoT device with an SE establishes a secure connection to the cloud to start exchanging information.

In general, the server and device will negotiate the cipher suite which contains TLS version, key exchange algorithm (i.e. ECDH, ECDHE, RSA), authentication algorithm (i.e. ECDSA, RSA), data encryption algorithm (i.e. AES_128_GCM) and message authentication code algorithm (i.e. SHA256) during the TLS handshake. Thus, the workflow for TLS establishment could be slightly different, depending on the agreement between client and server. The figure takes TLS1.2 ECDSA authentication algorithm and the ECDHE (Elliptic curve Diffie-Hellman ephemeral) key exchange algorithm as an example to illustrate what needs to be done in the VaultIC408 and how the mutual TLS handshake works for establishing a secure connection to the cloud application.

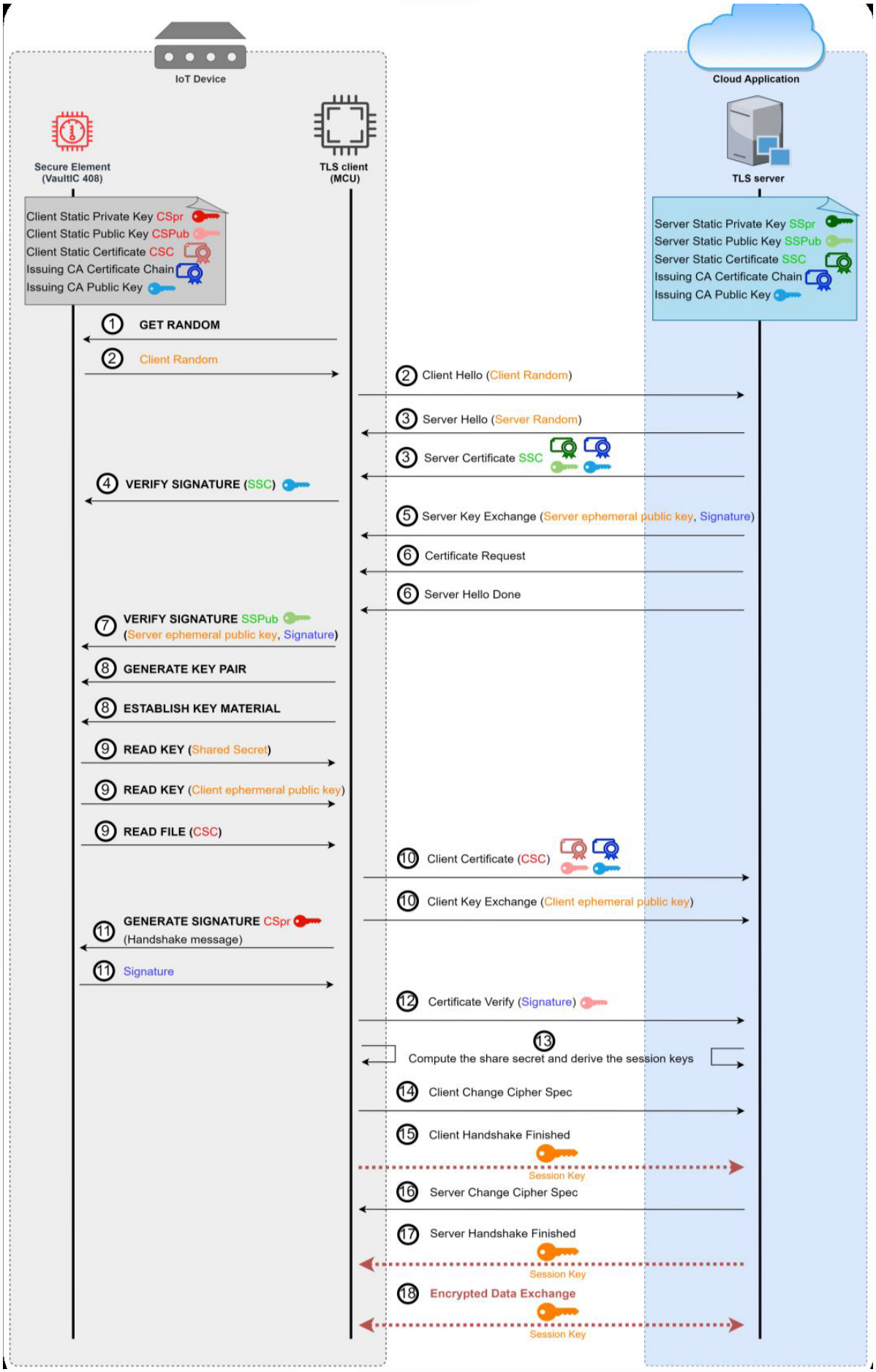


Figure 3: Establishing a secure connection to a cloud using TLS 1.2

1. Before the TLS client (MCU) issues the TLS connection request to the TLS server (Cloud), it asks the SE for a client random number by a GET RANDOM command.

2. The TLS client takes the random number and initiates the handshake by sending a Hello message to the TLS server. The message includes which TLS version, session ID, cipher suites supported by the client and a string of random bytes known as the "client random" which was previously generated by the SE.

3. The TLS server responds with the server hello message with the "server random" which is a random string of bytes generated by the server, followed by the server's certificate.

4. The TLS client issues the request VERIFY SIGNATURE command to SE in order to verify the authenticity of the certificate received from the TLS server with the public key of the issuing certificate authority that signed the server certificate. (Best practice is to store this ICA public key in the SE during personalization of the device, as to protect it from being modified).It confirms that the server is who it says it is, and allows the TLS client to keep interacting with the authentic the TLS server.

5. The TLS server takes the client and server randoms, DH domain parameters as well as server's ephemeral key pair that will be used to compute a shared secret. The TLS server signs them with its private key and sends the sever key exchange message with ephemeral public key and the signature to The TLS client.

6. The TLS server issues the certificate request and concludes with a "Server Hello Done" message to the TLS client.

7. The TLS client uses the public key from server's certificate to verity the digital signature of the Server Key Exchange message and checks if the server is the rightful owner of the key pair by VERIFY SIGNATURE command. If yes, it will start requesting the required keys to SE.

8. The TLS client issues the requests to SE for generating client ephemeral public key and establishing key material in order to share it with The TLS server for calculating the shared secret.

9. The TLS client requests the SE to compute the shared secret, which is calculated from server ephemeral public key and client ephemeral private key in the SE. Then, The TLS client reads client ephemeral public key and client certificate by READ KEY and READ FILE commands.

10. The TLS client sends the client certificate, issuing CA certificate chain and client ephemeral public key to the TLS server so that the TLS server can verify the signature of client certificate by the public key of issuing CA, and also calculate the shared secret from its own server ephemeral private key and the client's ephemeral public key in the next steps.

11. The TLS client issues the request to SE for generating the digital signature by using its private key so that the TLS server can verify if the TLS client is the authentic owner of the device certificate.

12. The TLS server verifies the signature by using public key from the client certificate.

13. Once the signature is verified, both sides derive the session keys from shared secret and the generated session keys will be used to encrypt/decrypt data.

14. The TLS client sends the Client Change Cipher Spec, which is used to change the encryption to the TLS server, at the point when the TLS client is ready to switch to a secure, encrypted connection. Any data sent by the client from now on will be encrypted by using the symmetric session key.

15. The TLS client sends Client Handshake Finished to the TLS server. It is the last message of the handshake process from the TLS client signifies that the handshake is finished and it's also the first encrypted message of the secure connection.

16. The TLS server sends Server Change Cipher Spec, meaning the TLS server is also ready to switch to an encrypted environment; any data sent by the TLS server from now on will also be encrypted by using the symmetric session key.

17. The TLS server sends the Server Handshake Finished message as the last message of the handshake process from the TLS server signifying that the handshake is finished.

18. Then, the secure session is established and all the data exchanges between the TLS client and server will be encrypted.

Solution architecture reference

Seal SQ is the only European company who can offer all the key components for the IoT from Certificate Authority (CA), Digital Identity, Certificate Management Solution (CMS), Secure Element (SE), Secure Identity Provisioning and Public Cloud Services Integration.

Below figure illustrates an example workflow of Seal SQ device-to-cloud solution for IoT, in a zero-touch provisioning architecture.

Please be aware that your application might require another bespoke architecture, depending on, e.g.:

- How do you define the digital identity to fit your IoT application;
- Do you prefer to pre-provision or post-provision the digital identity in the IoT device;
- Do you use an SE and what needs to be stored in it.

Seal SQ would provide the best solution architecture that could be applied to your IoT application and you can focus on the development in your domain.

Check the diagram below:

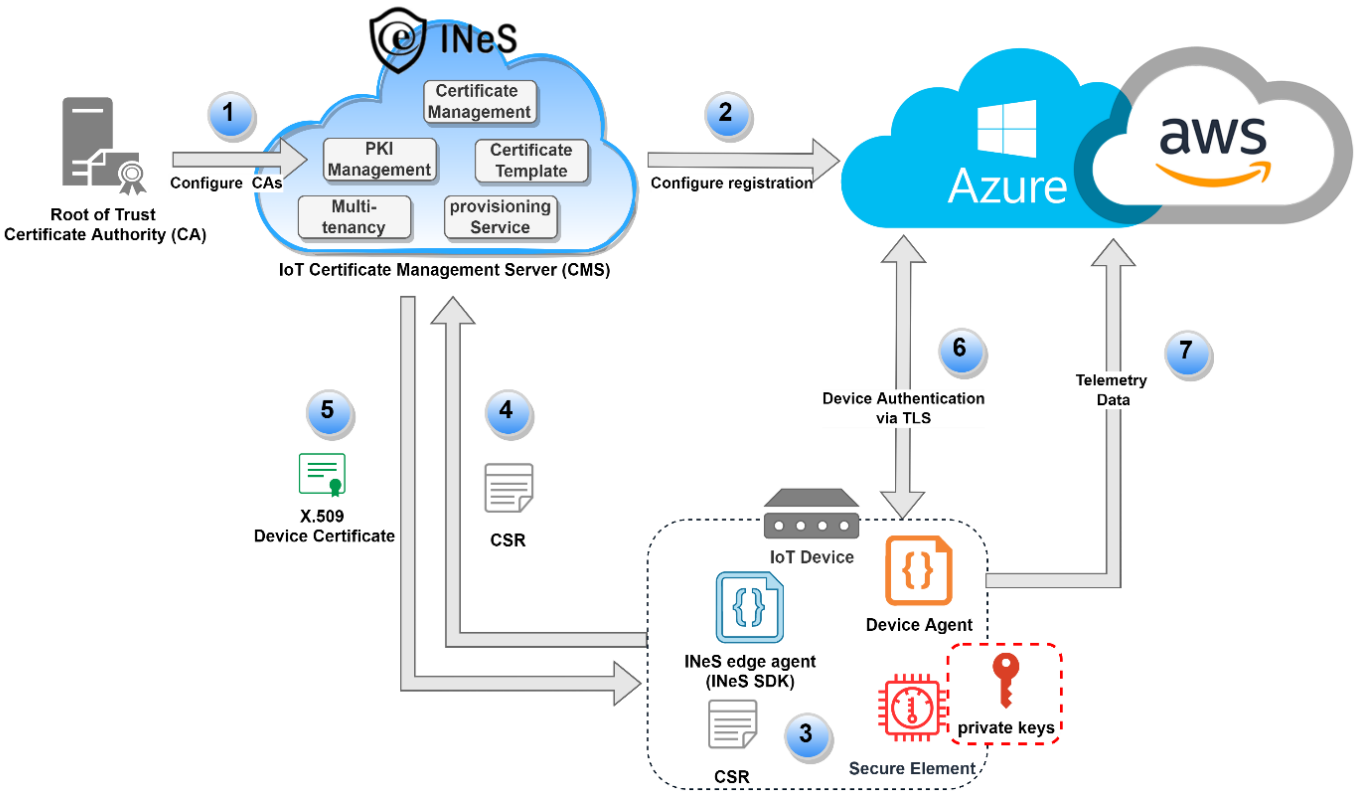


Figure 4: Zero-Touch device Provisioning example

1. Pre-configure the CA/ICA (issuing CA) for your organization on the INeS platform.

2. Pre-configure the public cloud connection in the INeS platform and register
the issuing CA through INeS CMS web portal to e.g. Azure DPS/IoT hub or AWS IoT core

3. The INeS client on the device generates the CSR which is signed by the pre-loaded private key that stored in the secure element.

4. The INeS agent requests for a device certificate from INeS by accessing the REST APIs and attaching the CSR.

5. Store the returned x509 device certificate in a specific directory of the IoT device.

6. The Device agent establishes a secure TLS connection to public clouds (AWS IoT Core/Azure DPS IoT hub) and on-boards the IoT device (the exact mechanism to on-board on these public clouds is out of scope of this document).

7. Once the IoT device is authenticated and on-boarded to the cloud, it starts publishing telemetry data to the messaging broker and the workload can be further analysed and processed by your IoT application.

Conclusion and key takeaways

IoT is disruptive technology that is growing rapidly. However, without proper security, it will never reach its full potential.

Device identity management is a key consideration to secure IoT devices—where the process must be protected for both authenticating devices and authorizing access based on permissions. One of the best ways to provision secure device identities is through a PKI.

Secure hardware provides the ideal trust anchor from which secure software and services can be leveraged:

1. A typical design pattern for OEMs wanting to build-in hardware security has been to solder a Secure Element (SE) chip alongside the primary System on Chip (SoC). Often these SE chips are delivered with the necessary secret keys and certificates so that they can be built into connected devices and on-boarded to popular cloud service providers such as AWS or Microsoft Azure. This pre-provisioning of keys simplifies the device manufacturer's production process and logistics considerably.

2. Most SEs also benefit from advanced protection from physical attack, which is an attractive capability to Cloud Service Providers that care about protecting their secret keys and the integrity of certificate data.

3. For Linux-based systems that use applications processor-based chips without integrated non-volatile memory, the SE also becomes a valuable place to store security variables between power cycles, e.g. lifecycle status or anti-rollback counters.

Using the Seal SQ PKI and Secure Elements provides a comprehensive, cost-effective, and scalable solution to multiple problems that the IoT industry is facing.

Acronyms and abbreviations

AES	Advanced Encryption Standard
CA	Certificate Authority, entity that signs digital certificates
CC	Common Criteria
CISO	Chief Information Security Officer
CMS	Certificate Management System
CSR	Certificate Signing Request
DDOS	Distributed Denial of Service
ECC	Elliptic Curve Cryptography, a public Key cryptography algorithm
ECDH	Elliptic-curve Diffie–Hellman
ECDHe	Elliptic-curve Diffie–Hellman ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
ICA	Issuing Certificate Authority
IoT	Internet of Things
JTAG	Joint Test Action Group: a JTAG interface can be used to access debug functions of a microcontroller and to access memory and registers
MCU	Micro Controller Unit
NIST	National Institute of Standards and Technology
OISTE	Organization for the Security of Electronic Transactions https://oiste.org/
PKCS#11	Public-Key Cryptographic Standards
PKI	Public Key Infrastructure https://en.wikipedia.org/wiki/Public_key_infrastructure
RBAC	Role Based Access Control
REST	Representational State Transfer
ROT	Root of Trust. The foundation for cryptography.
RSA	Rivest Shamir Adleman, a public Key cryptography algorithm
SaaS	Software as a Service
SHA	Secure Hash Algorithm
SHA	Secure Sockets Layer. Secure transportation protocol replaced by TLS
TLS	Transport Layer Security. A secure transportation protocol

References

[NIST] NIST SP 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation, January 2018

[FIPS] NIST FIPS 140-3: Security Requirements for Cryptographic Modules, March 2019
[CC] CC:2022 Release 1

[VIC408] Seal SQ: VAULTIC408 Summary Datasheet, March 2022

[AWS] Device Manufacturing and Provisioning with X.509 Certificates in AWS IoT Core

Disclaimer

Information in this document is not intended to be legally binding. Seal SQ products are sold subject to Seal SQ Terms and Conditions of Sale or the provisions of any agreements entered into and executed by Seal SQ and the customer.

The products identified and/or described herein may be protected by one or more of the patents and/or patent applications listed in related datasheets, such document being available on request under specific conditions. Additional patents or patent applications may also apply depending on geographic regions.

For more information, visit www.Seal SQ.com

Release date: December 2022

Contacts

Seal SQ SA
SEAL (BVI) Corp. Craigmuir Chambers, Road Town, Tortola, VG 1110, British Virgin Islands

Tel : +33 (0)4 42 370 370

Fax : +33 (0)4 42 370 024

Email: sales@Seal SQ.com

Stay connected with @SealSQ