WISeTrustBoot

Or How to Trust your IoT Device

Whitepaper

Contents

The need for trust

If the Internet of Things stands up to its promise, within several years from now, according to specialists, more than 50 billion devices will be connected and communicating over the Internet.

The data collected from or sent to these devices allows smart systems to control their environment and to take sound decisions, like sending in the right amount of water to humidify a soil to grow beans, checking the vital data of a patient at a remote location to take immediate action if a medical urgency arises, predicting maintenance for a specific engine before it fails or simply monitoring the process of thermonuclear reaction in a power plant. In any case, data exchanged with these devices will have more or less economic, legal or physical impact.

Today's technology gives us the possibility to connect in a cheap and easy way whatever device to a central system and, excited by the new opportunities this brings, most organizations keep on rapidly adding IoT[1] devices in a most economical way, without considering potential security threats. Attackers however, facing this new playground of easy accessible, non-protected, huge computing infrastructure, are well aware of the brand new opportunities it brings to perpetrate large scale attacks. IoT botnets such as Mirai, Brickerbot or Hajime[2] and their intensive DDoS attacks, showed that the Internet of Things is far from being exempt from cybercrime.

It is therefore not only necessary to secure the data transmitted to or from a device, or to authenticate the device before accepting a connection, but one has to ensure that the function of the device is unaltered and not compromised by any malicious piece of software. One has to establish a Chain of Trust, from device firmware up to the contents of the data communication, in order to be able to rely on the system and the information that is exchanged.

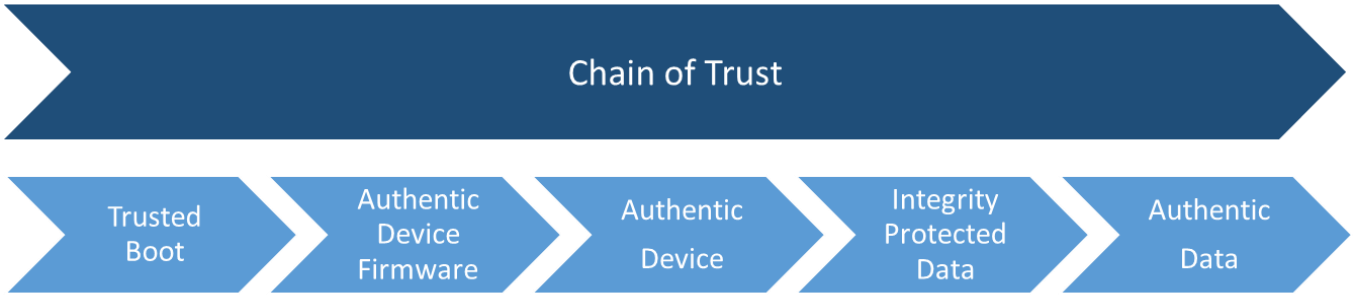


Figure 1 Chain of Trust

Data authenticity is a result of an authentic device firmware, running on an authentic device with an authentic digital identity, generating data that is protected against modification.

[1] All Abbreviations and Acronyms are explained at the end of the document
[2] See insert further down the document

From the above, it should be clear that we can only trust the data exchanged with a device if we have trust in the authenticity of that device and its firmware.

Therefore, WISeKey developed **WISeTrustBoot**, to keep your device cheap and easy, but trustworthy and secure.

ZOOM ON

Some examples of Malware

Mirai

Mirai is a malicious piece of software code that, once installed on a Linux system, transforms the system into a so called Bot. Mostly targeted to low cost consumer devices such as video cameras, the large number of Bots, operating in a Botnet, are used to attack distant systems. Mirai was used for large scale attacks in September 2016 against the French hosting service OVH and in October against DYN, a US based DNS service, provoking outage of several web services during some days.

Brickerbot

Brickerbot was a malware that once infecting a device started to destroy it by deleting the data in storage and disconnecting it from the network. An alert was raised against Brickerbot on April 2017.

Hajime

Hajime is a malware roughly identified at the same time as Mirai, but instead of the latter, is not intended for DDOS attacks. In fact, apart from closing some ports in the device avoiding infection by Mirai it does not seem to do anything else. Researchers do not know the intentions of Hajime but fear that in future the bots will be activated for a malicious goal.

Spectre and Meltdown

Discovered in early 2018, these two exploits use vulnerabilities in controllers which allows a malicious process to read data that is stored in 'secured' memory from another process. This data includes private keys, passwords and other sensitive data. It is related to a vulnerability in the hardware architecture of a majority of microcontrollers from big payers and patches have been made available by Microsoft, Google, Mozilla and other editors, slowing down the operation of the affected hardware by sometimes more than 30%.

It shows that it is a bad idea to use hardware that stores sensitive assets and in the same time provides an environment to run firmware and applications, because sooner or later, hackers will find vulnerabilities in the logical fence and infect your device.

Trusted boot

The starting point of the Chain of Trust introduced above, is called the Boot. The Boot is the piece of firmware between the hardware and the application firmware. It is executed just after the hardware is powered up when the device is still relatively dumb, starting up all resources of the device, bringing it alive step by step. It takes care of finally loading and executing the application firmware.

Now, in order to ensure the functionality of the device, the Boot must be able to check the integrity and authenticity of the application firmware before turning control over to the application. This implies that the Boot itself must be trusted. It is often called the Root of Trust of the device.

One way of making a Boot trusted is to store it in ROM, which makes it immutable. It executes initial verifications using embedded algorithms and secret keys, all stored in ROM and possibly executed in specially designed secure execution environments in the chip.

This only works for a specific hardware platform and is per definition a very static process, based on the same secret assets for every device and by consequence more interesting for attacks.

WISeTrustBoot is a solution allowing device makers to build a solid, secure Chain of Trust on essentially every type of microcontroller platform, regardless of the protection mechanisms the platform proposes.

Security needs of IoT Devices

The overall goal of trusting a device and its data as well as preventing the execution of non-desired firmware on it, pins down to some basic security requirements that are listed in the following sections.

But before going to the details of these requirements, the required knowledge, equipment, cost and complexity of executing an attack and the possible impact should be taken into account:

Category of attacks

- **Remote Attacks**: these attacks are executed via the Internet connection of the device. It requires knowledge of some weaknesses of the device, its operating system or its applications. Attacks can be relatively easy and cheap to execute, but very effective on a large scale. Think of Botnets such as Mirai, or attacks such as Spectre and Meltdown[3] that have cost the world up to 30% system performance, days of outage and several million dollars to the industry.
- **Board Level Attacks**: here the attacker needs physical access to an operational device. One could try to dump memory, spoof communication busses, analyze power consumption, decompile program code and the like. It requires very specific knowledge and equipment. The attack is executed on one device, and the retrieved information is not necessarily usable to put other devices at risk, but could result in a global compromise of all similar equipment, if countermeasures have not been implemented as they should.
- **Chip Level Attacks**: These are a very specialized types of attack, requiring very exclusive and expensive equipment and knowledge, aimed to obtain the secrets that are buried deeply in the silicon chips, or modify their behavior (Focused Ion Beam, Laser, Probing, decapsulation …[4]). As with the board level attacks, countermeasures against these attacks should avoid that the information recovered with the attack on a few devices is useful for an attacker to execute a large scale attack.

[3] Refer to insert elsewhere in this document
[4] Refer to insert elsewhere in this document

ZOOM ON

Chip Level Attacks

Focused Ion Beam

A FIB system is similar to a Scanning Electron Microscope, with the difference that it does not use a beam of electrons, but rather Gallium Ions. Depending on the strength of the beam current used, one can either create images or cut or replace material inside the substrate. Generally used by circuit designers to redirect signals, rework trace paths and add or remove components, FIB can be used by attackers to remove security features and change the functioning of sensitive operations within a chip. FIB systems are very costly and their operation requires highly skilled personnel, so attacks are generally dedicated to high value targets.

Laser

Lasers can be used to remove layers from a chip or to cut metal tracks within a chip, to probe active regions or to inject faults within the chip by optical flashing at the right moment. It requires skills and equipment, but to a lesser extent than FIB.

Clock or Power Glitches

This form of attack injects glitches in for instance the clock frequency or the power supplied to the chip, with the aim to disturb its operation. It can result in incorrect instruction fetch or incorrect EEPROM data read.

Decapsulation

Decapsulation is a first step into a chip level attack. Using strong chemicals such as fuming nitric acid, the package of the chip is (partly) removed, so the silicon becomes accessible for imaging or probing with other equipment.

Security needs

Protection of credentials and secrets

The credentials used by the device to authenticate, e.g. a private key of a digital identity, or the secrets, like a shared key to decrypt information, need to be protected at least against Remote Attacks and Board Level Attacks. Depending on the sensitivity of the application, Chip Level Attacks should be taken into account as well.

WISeTrustBoot accomplishes the protection of the credentials and secrets by using a tamper resistant Secure Element.

This is a specific microcontroller, designed exclusively for this goal. The Common Criteria[5] certification attests the level of protection against Chip Level attacks. See the section further down explaining the characteristics of WISeKey's tamper resistant chip VaultIC, a Secure Element complying with this certification.

[5] The Common Criteria for Information Technology Security Evaluation (CC) is the technical basis for an international agreement, the Common Criteria Recognition Arrangement (CCRA), which ensures that products can be evaluated by competent and independent licensed laboratories so as to determine the fulfilment of particular security properties, to a certain extent or assurance.

Device integrity

As mentioned in previous sections, the integrity of a device is essential for a successful IoT system. The integrity is obtained through a chain of validations, starting from the trusted Boot up to the highest level application firmware, where every validated firmware component verifies the next level component.

WISeTrustBoot ensures device integrity starting with the Boot code. It is executed from a protected partition of the Non Volatile Memory of the device. The protection consists of the standard mechanisms proposed by Microcontroller Unit (MCU) of the device, like disabling JTAG and restricting Read/Write access rights. The Boot code executes a Hash of the firmware and uses the services of a Secure Element, called VaultIC, to validate it against the signed hash[6] received during initial provisioning or subsequent update of the device.

In order to ensure protection of this process, the Boot code starts with establishing a secure channel (SCP03, ref [SCP]) to the Secure Element, resulting in a protected communication between Boot code running on the main MCU and VaultIC; all commands and responses being encrypted and authenticated throughout the process, thus protecting the physical link between MCU and VaultIC against eavesdropping, data modification or substitution.

Essential information for the integrity check, like key material, are stored in and exclusively used by the Secure Element, while sensitive application information , is protected in the Secure Element VaultIC.

Device update authenticity

New Firmware shall only be accepted by the device (in fact, by the Boot loader) if its authenticity can be established. This is done by WISeTrustBoot similarly to the initial device integrity check, see the illustration below:

When a new firmware version is ready for distribution, it will be encrypted and signed by the official developer, using a code signing tool called SignaTool, delivered with WISeTrustBoot. First, the genuine firmware code is encrypted using the AES algorithm with a secret, application specific key. Subsequently the hash of the firmware is signed by the developer. The secret assets for this, i.e. the private key and encryption key are stored on a USB-stick with a tamper resistant chip VaultIC. The identity of the genuine developer is proven by a certificate from the Trusted Root Certification Authority OISTE WISeKey Global or another Certification Authority.

At reception in the device, WISeTrustBoot verifies the version number of the new firmware against the previous version number stored in VaultIC. Then the firmware is hashed and the hash is validated by VaultIC against the signature sent with the new firmware.

This certificate is stored in the VaultIC of the device, hence it cannot be replaced by a rogue certificate that could allow third party developers to sign malicious code with a fake signature.

[6] A hash is a cryptographic operation transforming data of an arbitrary length into a result of a fixed length. The properties of the operation are such that even the slightest modification of the original data results in a completely different hash value. A hash is not reversible. It is impossible to find the original data from the hash value.

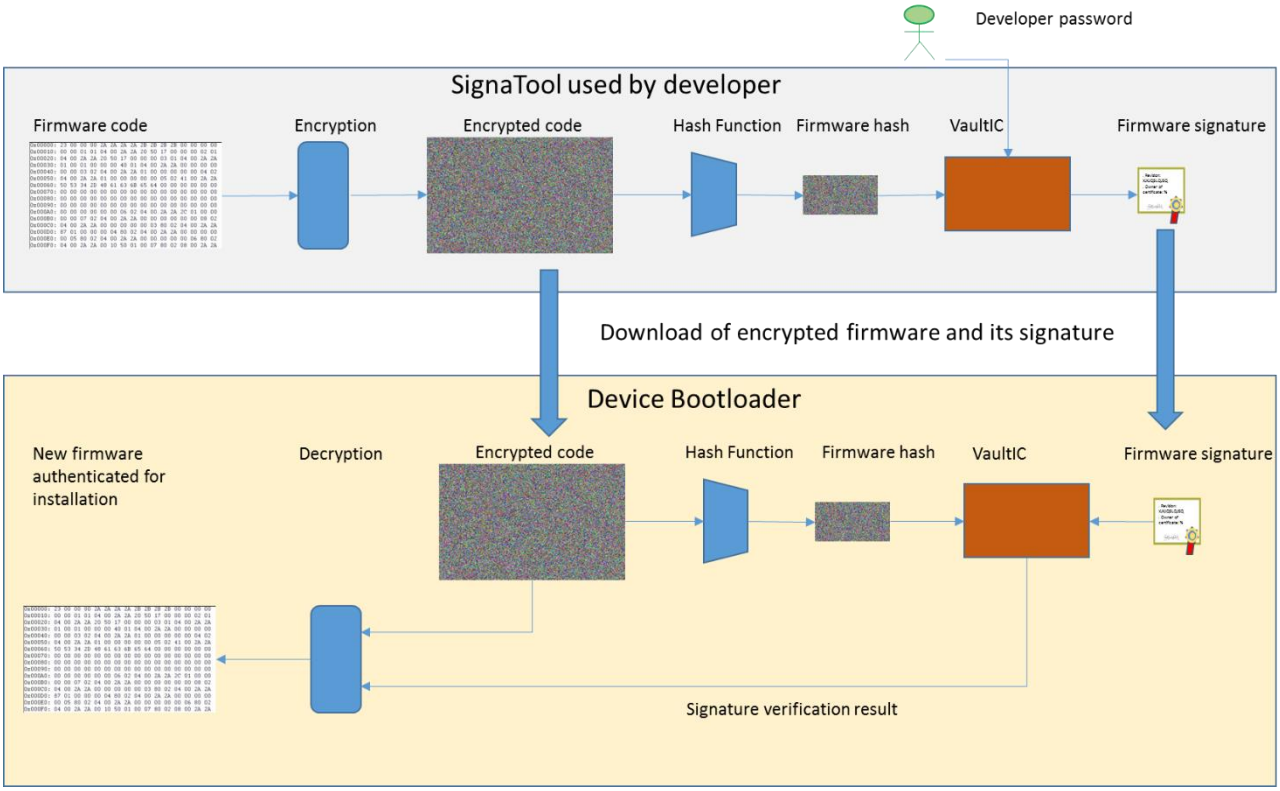


Figure 2 Verification of the authenticity of a new firmware version by WISeTrustBoot

Only when the signature is declared valid by VaultIC and the version number is approved, the new firmware is decrypted with a key stored in VaultIC and installed on the device.

Protection against counterfeiting or cloning

Ever imagined the impact counterfeited devices in the field? There first of all the loss of revenue due to the decrease in sales of genuine devices. Then there is brand image attrition due to quality or after-sales issues, including cyber-attacks perpetrated with fake devices that are perceived by general public as being originated from original devices.

VaultIC stores and operates on secret information such as authentication keys, on behalf of the main MCU of the device. And as WISeTrustBoot binds VaultIC to the main MCU with the secure channel protocol SCP03, the device can only operate with this specific VaultIC being present.

Using another chip (even if it is another VaultIC), or no chip at all will result in corrupted or no data exchange for essential information and the device will not be able to operate. Further, to check the presence of VaultIC, a third party (mostly the back-end service using the data from the device) can execute a remote authentication of the device. It consists of having a random challenge signed with a private key that is securely stored in VaultIC (the necessary functions for device authentication are available through the API proposed by WISeTrustBoot). Because of its tamper resistance, this key is deemed not clonable. Therefore a correct signature of the challenge proofs the presence of VaultIC, hence the authenticity of the device.

Protection of data at rest

One level of protection of the device against board or chip level attacks is to avoid that sensitive data can be dumped from an unpowered device and used for further attacks.

This protection is accomplished by WISeTrustBoot by storing the most sensitive data like authentication keys in VaultIC, the tamper resistant secure microcontroller. Further sensitive data is stored encrypted in memory of the main MCU and protected against dumping with the protection mechanisms proposed by this MCU.

Protection of data during execution

In general it is easier to discover sensitive information during its use by the host controller, than when it is at rest, stored somewhere in memory. It is for instance a matter of minutes to discover a secret key used in a poorly coded AES encryption, if the right equipment is used and a sufficient number of encryption rounds is analyzed. Techniques used here are e.g. Differential Power Analysis or Correlated Power Analysis.

WISeTrustBoot is designed and coded to be resistant against these kind of attacks. It is therefore merely impossible to extract secret information from the Boot loader during execution of the code.

Protection of communication

Once the device is up and running an authentic application, the communication with the back-end service needs to be secured as well, to prevent an attacker from tampering or spoofing the data.

VaultIC proposes a full set of functions, made available through the API in WISeTrustBoot, allowing an application to build a secure communication channel, e.g. TLS/SSL or similar.

WISeTrustBoot components

WISeTrustBoot consists of three major components:

- the Secure Bootloader binary on the device MCU,
- VaultIC407 in the device and
- SignaTool, a PC tool for the firmware developer

These components are described in the following sections.

Secure Bootloader Code Binary

WISeTrustBoot is delivered as a binary file compiled for the desired platform. It consists notably of a cryptographic component executing side channel resistant[7] algorithms and an SCP03 component, establishing a secure binding with the VaultIC.

Furthermore it consists of the Secure Bootloader function itself that operates the trusted Boot and secure firmware download as described in this document (sections Device integrity and Device update authenticity). The Application can use the API of the Boot loader to request firmware update and to use the powerful VaultIC407 toolbox for device authentication or other application specific cryptographic needs.

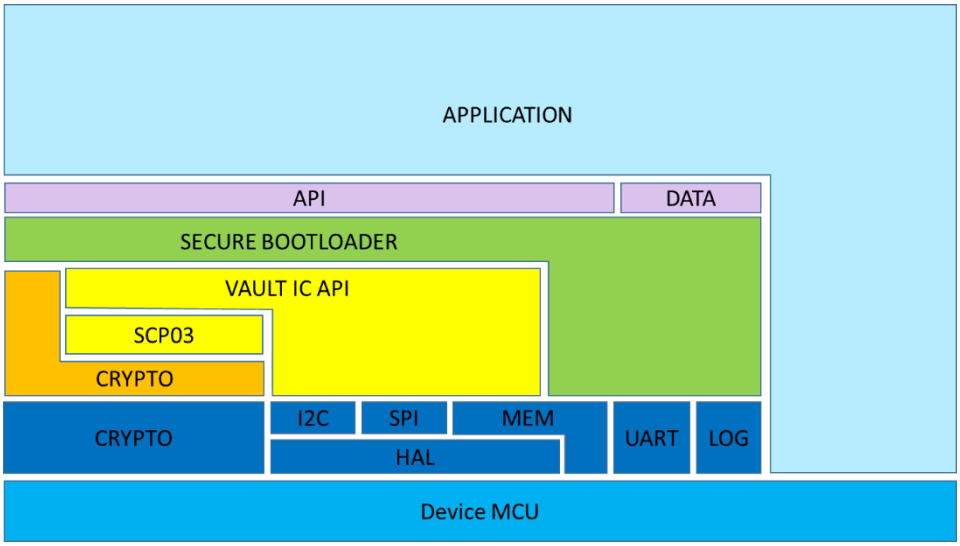


Figure 3 WISeTrustBoot Architecture

VaultIC407

WISeTrustBoot uses VaultIC407 to store the secure assets, like digital certificate and associated device private keys, or critical application specific data. VaultIC407 is a Secure Element, existing specifically for the purpose of performing its pre-programmed security routines. These cryptographic services and

[7] Side Channel proof algorithms are designed in a way that the execution of the cryptographic calculations is resistant to side channel attacks, i.e. it is very hard if not impossible for an attacker to discover a secret key by observing the power consumption or electromagnetic radiation of the device during operation.

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 configurable cryptographic tool boxes for Authentication, Confidentiality and Integrity executed in a secure environment. The tool box proposes a variety of standard and NIST[8] 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 to guarantee the entropy needed for high-level cryptographic services.

Secure Elements, such as WISeKey's VaultIC, provide multiple advantages when compared with software-based security, including:

- Crypto keys and other security materials are stored under control of secure hardware, beyond the reach of any software attack.
- 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.

VaultIC are certified by the US FIPS 140-2 Level 3 (version 2011) standard and Common Criteria EAL4+/5+.

SignaTool

The verification of the application firmware is based on the verification of the signature of the hash of the code. The hash is calculated by the authentic publisher of the code and cryptographically signed, using a private key stored on a USB-Stick containing a Tamper resistant chip from the VaultIC family.

On the device side, before starting the application, the Bootloader calculates the hash as well and sends it to VaultIC407, together with the received signature. VaultIC407 verifies the signature using the public key of the publisher, contained in the certificate stored in its secure memory.

SignaTool consists of

- A PC tool: using a powerful graphical user interface or command line interface, the application developer can build the image of the application firmware destined to the device, compute a hash and subsequently sign it with his or her private key, to proof the authenticity and integrity of the firmware.

8 National Institute of Standards and Technology

- This private key from the developer is securely stored on a USB stick containing a VaultIC Secure Element. The VaultIC in the stick generates the signature with the Elliptic Curve Digital Signature Algorithm ECDSA. The USB stick itself is protected with a password, to prevent unauthorized developers to use the key to sign an application.

Conclusion

WISeKey's WISeTrustBoot solution, is the first platform-independent Secure Boot and Secure Firmware Update solution that combines the strength of tamper resistant secure elements, state-of-the-art crypto libraries and strong digital signatures. It provides an easy and cost effective solution to protect your IoT device against the ever increasing risk of cyber-attacks.

Acronyms and abbreviations

AES	Advanced Encryption Standard
CA	Certificate Authority, entity that issues digital certificates
CC	Common Criteria
DDOS	Distributed Denial of Service
ECC	Elliptic Curve Cryptography, a public Key cryptography algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
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
OISTE	Organization for the Security of Electronic Transaction https://oiste.org/
PKI	Public Key Infrastructure https://en.wikipedia.org/wiki/Public_key_infrastructure
PKCS#11	Public-Key Cryptographic Standards
ROT	Root of Trust. The foundation for cryptography.
RSA	Rivest Shamir Adleman, a public Key cryptography algorithm
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer. Secure transportation protocol replaced by TLS
TLS	Transport Layer Security. A secure transportation protocol

References

SCP	Global Platform. Secure Channel Protocol 03 - Card Specification v2.2 Amendment D - February 2009

About WISeKey

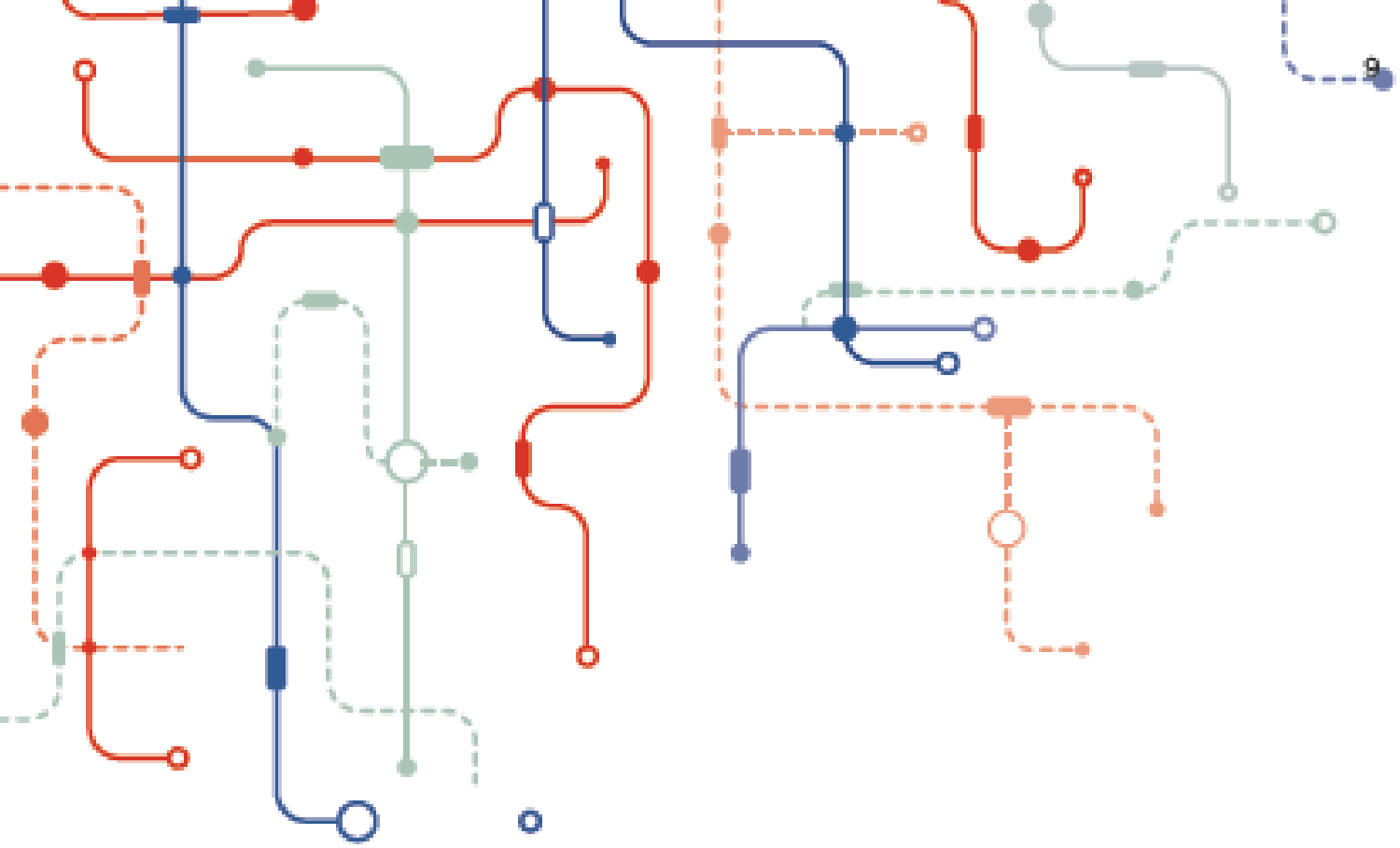
WISeKey is a leading global cybersecurity company offering since more than 20 years tailor made security solutions for devices, website, people and services, based on a scalable Public Key Infrastructure.
WISeKey Semiconductors secure the pervasive computing shaping today's Internet of Everything.
WISeKey has an installed base of over 1 billion tamper resistant chips in virtually all IoT sectors (connected cars, smart cities, drones, agricultural sensors, anti-counterfeiting, smart lighting, servers, computers, mobile phones, crypto tokens etc.).

Our technology, trusted by the OISTE/WISeKey's Swiss based cryptographic Root of Trust ("RoT") provides secure authentication and identification, in both physical and virtual environments, for the Internet of Things, Blockchain and Artificial Intelligence. The WISeKey RoT serves as a common trust anchor to ensure the integrity of online transactions among objects and between objects and people.
Our certificate Authorities, Security Brokers, management systems and tamper resistant secure microcontrollers are regularly audited and accredited with highest grade Webtrust and Common Criteria certifications.

Disclaimer

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

For more information, visit www.wisekey.com



Contacts

WISeKey SA
WTC II
29, route de Pré-Bois
1215 Geneva • Switzerland
Tel: + 41 (0)22 594 30 00
Fax: +41 (0) 22 594 30 01

WISeKey Semiconductors
Arteparc Bachasson • Bât A
Rue de la carrière de Bachasson
13590 Meyreuil • France
Tel : +33 (0)4 42 370 370
Fax : +33 (0)4 42 370 024

www.wisekey.com
Stay connected with @WISeKey