Security Primitives:
Common Nomenclature to Describe Security Requirements in (I)IoT Systems

Table of Contents

1 Introduction

Devices in the Internet of Things (IoT) and Industrial IoT (IIoT) need to be protected against cybersecurity threats. Hardening these devices and protecting the personal assets of end users has become a significant focus of system designers, developers, and manufacturers[1, 2], as well as regulators and legislation[3, 4].

There do not exist two documents on securing (I)IoT devices that seem to agree on a common definition of these devices or even a common terminology to describe the security requirements. Furthermore, the scope of such documents depends on whether they relate to certification, legislation, or implementation guidance.

With the topic of security becoming relevant for a broad audience of implementers, a common terminology to agree, understand and implement measures to fulfill standards and protect against security threats is getting increasingly important.

1.1 Objective

This document aims to establish a common vocabulary to describe security requirements in (I)IoT systems. It introduces a number of "security primitives" by distilling common terms out of various standards to describe non-overlapping security features on multiple levels—from rather low-level platform features such as software isolation to high-level functionality such as secure updates.

Furthermore, this document describes a process to identify relevant requirements for an (I)IoT system out of a use case description of the system. A map to existing standards, certification schemes, legislation, and popular implementation guides is provided, which allows for quickly identifying implementation requirements for an (I)IoT product.

The security primitives and the related process are intended as an entry point for gathering security functional requirements and process requirements for a particular use case.

1.2 Scope of this Document

(I)IoT describes an ever-growing variety of consumer, home, and industrial devices with network connectivity. These devices are interacting with the physical world through a transducer, i.e., a sensor or actuator, and incorporate at least one network interface[5]. As with most sources defining (I)IoT, conventional information technology (IT) devices such as personal computers, laptops, smartphones, or tablets are explicitly excluded from the range of (I)IoT devices.

While most sources broadly agree on this definition of (I)IoT devices, there is no consensus on the scope of security requirements. Regulators and legislation typically evaluate this topic from the end user perspective and consequently target the full (I)IoT ecosystem, including the devices, the cloud backend, and everything in between. Component manufacturers and most certification schemes, on the other hand, typically target (I)IoT devices or components thereof.

The scope of this document is an (I)IoT system as depicted in Figure 1. This system consists of one or more (I)IoT devices and the cloud backend to which they are connected. It also includes all processes related to all stages of the device life cycle, such as designing and manufacturing, as well as operating the devices and the cloud backend. In this scope, an (I)IoT product is defined as an (I)IoT system comprised of one (I)IoT device and the accompanying cloud backend.

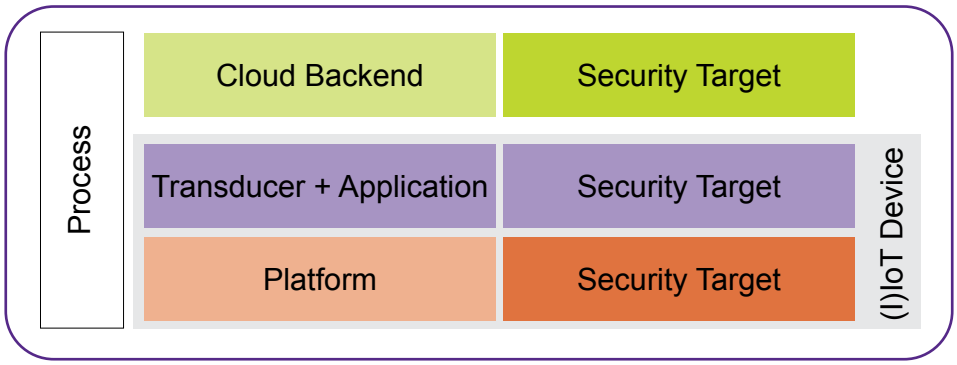

Figure 1: Definition of an (I)IoT System and an (I)IoT device

The (I)IoT device is further broken down into the security platform part that typically consists of a secure microcontroller or microprocessor unit (MCU/MPU) on a system on chip (SoC), and the (I)IoT application running on top of it. The platform part may also contain companion chips such as secure elements, as well as the as well as the firmware, operating system, device drivers, and software stacks, enabling secure operation on the MCU/MPU. The (I)IoT application part contains the sensors and actuators it requires for its operation.

This split of the device roughly resembles the split of what a platform/chip manufacturer provides to enable its customers and the specific functionality an (I)IoT device manufacturer implements.

The cloud backend includes the backend application as well as the infrastructure required to connect the (I)IoT device to the backend. Especially in the industrial domain, this includes equipment such as routers, switches, and firewalls. Devices such as hubs that sit on the edge and connect devices to the cloud are also part of the cloud backend by this definition.

Throughout this document, the following actors are identified:

- Manufacturers—provide the platform on which the (I)IoT devices are built; they design and develop SPUs, components and related enablement kits
- Original Equipment Manufacturer/Original Device Manufacturer—the primary customers of the manufacturers; they design, develop and operate (I)IoT devices and related services
- End users—the intended consumers, operators or system integrator of (I)oT devices and their services in the industrial or consumer electronics market

This document is intended as an entry point for gathering security functional requirements and process requirements for a particular use case. It defines a process to evaluate use cases against common standards such as ISA/IEC 62443[6] and select an appropriate platform. It helps (i) to identify gaps in general functionality, (ii) to analyze how secure components support a full system solution, (iii) to discuss the security level and attack robustness level needed for a primitive. It does not provide implementation requirements or grant any security claims. A detailed security analysis of the individual security functional requirements and their implementation is strongly recommended as a subsequent step. While this process does provide guidance towards fulfilling certification requirements, it is not certification evidence by itself but provides a structure that helps to create evidence and rationale for certification.

Section 2 provides an overview of the security primitives as a common vocabulary to describe security requirements in (I)IoT systems. This is followed by a process description to apply these primitives to use cases and products in Section 3. A detailed description of the individual primitives and their inter-dependencies is provided in Section 4 for the security-functional primitives and in Section 5 for the process-related primitives. Finally, Section 6 gives an outlook on the next steps and future extensions of the proposed methodology.

1.3 Intended Audience

The intended audience of this document includes engineers and developers working on (I)IoT systems from platform to backend level, as well as certification and compliance engineers, managers, decision-makers, and everyone interested in a system view on (I)IoT security. Reading this document does not require any expert knowledge on security but is intended to provide a common understanding of terminology (to follow the requirements of relevant standards and implement measures against security threats.)

1.4 Primitive Derivation Methodology

To derive a common vocabulary, many different sources were collected. The sources include requirements and terminology from legislation, standards and recommendations, and criteria from evaluation and certification methodologies.

Additionally, the requirements of the following standards are considered:

- ISA/IEC 62443 4-2: "Security for Industrial Automation and Control Systems"[6]
- ETSI EN 303 645: "Cyber Security for Consumer Internet of Things"[8]
- SAE J3101: "Hardware Protected Security for Ground Vehicles"[9]
- FIPS 140-2[10] and FIPS 140-3[11]: "Security Requirements for Cryptographic Modules"

Requirements from legislation such as the following are considered:

- The United States IoT Bill[4]
- United Kingdom Government Code of Practice[3]
- Finish Cybersecurity label[7]

To provide a mapping to common certification schemes and certification methodologies, criteria of the following sources (amongst others) are considered:

- Security Evaluation Standard for IoT Platforms (SESIP)[2]
- GlobalPlatform Trusted Execution Environment (TEE) Protection Profile (PP)[12]
- Arm® Platform Security Architecture (PSA) Level 2 and Level 3[13]
- GlobalPlatform IoTopia[1]

Finally, the terminology of the following recommendations is collected:

- NISTIR 8259: "Recommendations for IoT Device Manufacturers: Foundational Activities and Core Device Cybersecurity Capability Baseline"[5]
- ST Microcontroller AN5156: "Introduction to STM32 microcontrollers security"[14]

From this input, a mapping table was created to build a vocabulary. The resulting categories were merged and distilled to find non-overlapping security features. These features are called "security primitives" in the remainder of this document. As a by-product of this derivation method, the derived security primitives are defined on multiple implementation levels and contain rather low-level product features such as software isolation and high-level functionality such as secure updates.

1.5 Terminology

Throughout this document, the keywords "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" are to be interpreted as described in "Key words for Use in RFCs to Indicate Requirement Levels"[15]. Additionally, the term "MANDATORY DEPENDENCY" denotes an "is required by" relationship between two entities, whereas "OPTIONAL DEPENDENCY" describes an "is utilized if present" relationship.

Finally, "CRYPTOGRAPHIC KEY MATERIAL" relates to cryptographic private, public, or shared keys or secrets, as well as cryptographic certificates and certificate chains.

2 Overview of the Security Primitives

As described in the introduction, security primitives constitute non-overlapping categories of security features, requirements, and terminology that provide a meaningful security service or functionality group. They provide a vocabulary to describe security features and requirements of an (I)IoT product and provide an easy mapping to platform features and certification requirements. The primitives relevant for a particular (I)IoT product could result in implementation requirements for hardware, software or even for appropriate processes. An overview of these primitives is given in Table 1.

Security Functional Primitives	Device Attestation Secure Updates Secure Onboarding and Offboarding Secure Provisioning and Decommissioning Secure Communication (Protocols) Secure Debug and Test Secure Backup and Recovery Account Authentication and Management (Attested) Secure State and Life Cycle Management Genuine Identification Secure Initialization Anomaly Detection and Reaction Cryptographic Key Generation and Injection Cryptographic Key and Certificate Store Secure (Encrypted) Storage Cryptographic Operation Cryptographic Random Number Generation System Event Logging Silicon Root of Trust Residual Information Purging Software Isolation Monotonic Time
Security Process Primitives	Secure Policy Compliance Security by Design Vulnerability and Incident Management Protection of Personal Information

Table 1: Overview of security primitives

The table is split into security functional primitives and process-related primitives. A more detailed description of each of the primitives is provided in an implementation-agnostic way in Sections 4 and 5.

First applications of the primitives to use cases and products in the industrial IoT sector, the Smart Home and Medical domains, as well as to security requirements in the Automotive domain, have been successful and consistent. This resulted in a process to evaluate use cases and products that guide a developer, engineer, or designer through the identification of security requirements of (I)IoT products. This process is described and applied to a simplified example use case in Section 3.

3 Process for Application to Product and Use Cases

This section defines a process to ease the application of the Security Primitives to use cases and products. As depicted in Figure 2, this process is divided into three distinct phases:

- The selection of applicable security primitives based on the use case
- Selecting an applicable standard and identifying relevant primitives (in some cases the applicable standard to be met might also be given as initial precondition)
- Choose a product/platform to realize the system and evaluate it against the security requirements to derive implementation requirements

The intention is to start with the top-most item and move to the bottom, but the order of the individual steps can be interchanged. Details for the individual stages of the process are given in the following subsections.

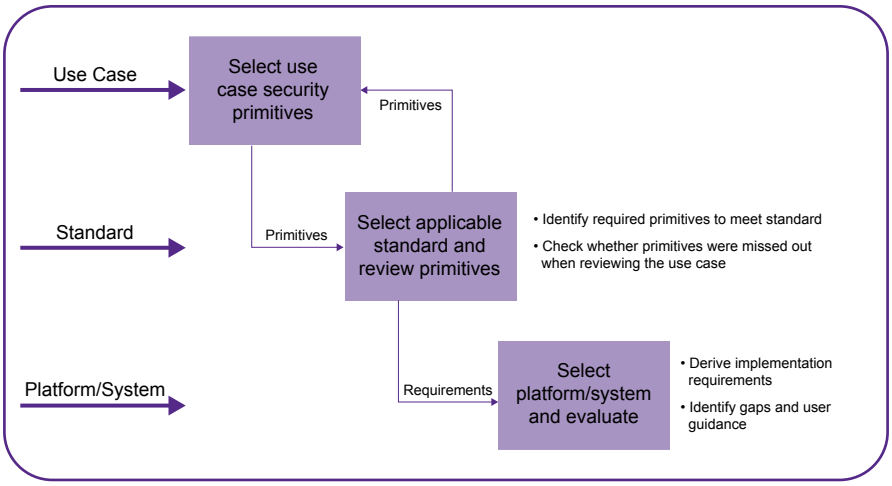

Figure 2: Standard evaluation for use cases and products

To illustrate the application of this process, a smart surveillance camera connected to the cloud is considered as a use case. Please note that the following sections are for illustration only and do not evaluate all requirements and primitives. They shall not be considered a complete analysis. Instead, this document focuses on the concern to securely connect to the cloud backend and to deliver an authentic video stream from the camera.

3.1 Use Case Evaluation Phase

As shown in Figure 2, the input to this phase is a use case description. From this definition, a list of applicable security primitives is defined.

Taking the example of the smart surveillance camera, the security primitive "Secure Communication (Protocols)" outlined in Section 4.5 is immediately applicable for the connection to the cloud backend.

However, this primitive has some dependencies, namely Cryptographic Operation (Section 4.16) and Cryptographic Random Number Generation (Section 4.17). These requirements are, therefore, indirectly applicable to this use case as well. Table 2 shows how these features are mapped to the security primitives.

Primitive	Applicability
Secure Communication (Protocols)	The smart surveillance camera needs to connect securely to the cloud using HTTP Live Streaming (HLS) secured via Transport Layer Security (TLS) v1.3.
Cryptographic Operation	*Implicit requirements from Secure Communication (Protocols):* • *Required cryptographic algorithms for TLS v1.3:* – *DHE-RSA* – *ECDHE-RSA* – *ECDHE-ECDSA* – *AES-GCM* – *AES-CCM* – *ChaCha20-Poly1306* – *HKDF-SHA256*
Cryptographic Random Number Generation	*Implicitly required by Secure Communication (Protocols).*

Table 2: Applicable security primitives to the example use case of a smart surveillance camera
**Implicit requirements are given in italics.

The full list of primitives given in the following sections provides guidance for the use case analysis. For the following phases, it is beneficial to detail the applicability and use case requirements as much as possible.

It is important to note that the use case evaluation phase only considers the (abstract) use case. Ideally, it does not include features of particular platforms or the whole system, and it does not impose any limitations or requirements of certain standards. As such, this step can be performed during a product conception phase and does not require platform specialists.

3.2 Standard/Certification Evaluation Phase

Once all primitives are evaluated with respect to the use case, a mapping can be performed to see which requirements arise from compliance to particular standards or regulations. For this purpose, a mapping table is provided for the documents listed in Section 1.4. This mapping needs to be applied to the analysis performed in the previous phase.

When applied to standards, regulation, or legislation this comparison yields two important results: on the one hand, it immediately results in security functional requirements (SFRs) the (I)IoT device needs to fulfill. On the other hand, some security primitives might not be mapped or even required as per the use case analysis but required by the chosen standard. For the example security primitives given in the previous section, the ISA/IEC 62443 4-2 requirements[6] for security level 3 (SL3) are listed in Table 3. These requirements need to be fulfilled by the (I)IoT solution.

Primitive	ISA/IEC 62443 4-2 SL3 Requirements
Secure Communication (Protocols)	CR 1.1.2 Multifactor authentication for all interfaces CR 1.2.1 Unique identification and authentication CR 1.8.0 Public key infrastructure certificates CR 2.2.0 Wireless use control CR 2.5.0 Session lock CR 2.6.0 Remote session termination CR 2.7.0 Concurrent session control CR 3.1.0 Communication integrity CR 3.1.1 Communication authentication CR 3.8.0 Session integrity CR 4.3.0 Use of cryptography CR 5.1.0 Network segmentation CR 7.1.0 Denial of service protection CR 7.1.1 Management communication load from component CR 7.6.0 Network and security configuration settings CR 7.6.1 Machine-readable reporting of current security settings
Cryptographic Operation	CR 1.8.0 Public key infrastructure certificates CR 1.9.0 Strength of public key-based authentication CR 1.14.0 Strength of symmetric key-based authentication CR 3.1.0 Communication integrity CR 3.1.1 Communication authentication CR 3.3.0 Security functionality verification CR 3.4.0 Software and information integrity CR 3.4.1 Authenticity of software and information CR 3.8.0 Session integrity CR 3.9.0 Protection of audit information CR 3.14.0 Integrity of boot process CR 3.14.1 Authenticity of boot process CR 4.1.0 Information confidentiality CR 4.3.0 Use of cryptography CR 7.3.1 Backup integrity verification
Cryptographic Random Number Generation	CR 2.12.0 Non-repudiation CR 3.1.0 Communication integrity CR 3.1.1 Communication authentication CR 4.3.0 Use of cryptography

Table 3: Resulting requirements for ISA/IEC 62443 4-2 SL3.

However, for the example of the smart surveillance camera, the security primitive "Secure Backup and Recovery" (Section 4.7) might not be relevant. This gap can be resolved in multiple ways: either it would require a modification of the use case (with a subsequent delta analysis), or it would require a tailored product certification with an argument outlining why certification can be achieved without fulfilling this requirement.

A similar mapping can be performed to certification schemes such as SESIP[2]. In this case, a list of building blocks for the certification is achieved. Further analysis is required here to investigate which of these are applicable to the (I)IoT product and the targeted security level.

Once this analysis is completed, a complete list of requirements with references to the relevant standards is available and can be handed over to the system evaluation phase. As with the previous phase, this phase is independent of the platform or system related to the (I)IoT device. Also, this phase does not need to consider the particular use case beyond the mapping provided in the previous phase.

3.3 System Evaluation Phase

In the final phase, the security primitives are mapped to platform and system features. The requirements of the previous phase can be mapped to concrete implementation details. This allows selecting the platform that best matches the use case and requirements, as well as identifies relevant software stacks and libraries. It also provides a list of implementation requirements and gaps that need to be covered by user guidance documents.

The resulting implementation security requirements out of this phase are purely functional at this point. A dedicated security analysis of the use case, the platform, and the market is still required to estimate the level of security hardening of the platform required on top. This, however, is not in the scope of this document. Please refer, for instance, to the SESIP methodology[2], which provides a toolbox for security certification on different security levels.

Returning to the example of the smart surveillance camera, the chosen example platform includes the NXP® i.MX RT1050 cross-over MCU for industrial products. The security features of this microcontroller are extracted from the data sheet[16] and given in Table 4.

Primitive	Security Feature of the i.MX RT1050
Secure Communication (Protocols)	
Cryptographic Operation	Data coprocessor (DCP): • AES-128, ECB, and CBC mode • SHA-1 and SHA-256 • CRC-32 Bus Encryption Engine (BEE) • AES-128, ECB, and CTR mode • On-the-fly QSPI flash decryption
Cryptographic Random Number Generation	True random number generation (TRNG)

Table 4: Security features of the i.MX RT1050, taken from the data sheet[16].

Comparing this to the applicable primitives in Table 2 and Table 3 shows that the SoC hardware itself can only provide partial functionality for TLS. Support for certain cryptographic algorithms such as elliptic-curve cryptography (ECC) or Rivest–Shamir–Adleman (RSA) that are required for TLS is missing. This gap could either be closed by choosing an appropriate software implementation that provides this functionality, or by augmenting the platform with a dedicated secure element such as the NXP EdgeLock™ SE050[17, 18]. Here, this particular platform has been chosen to highlight that the system evaluation phase may yield gaps. For this particular use case, one might rather choose an SoC with hardware support for the required cryptographic functionality, such as one of the LPC55S69 security solutions for IoT[19].

4 Definition of Security Functional Primitives

This section covers functional security primitives of (I)IoT systems. These primitives are defined in an implementation-independent way, and their inter-relations are highlighted. Not all primitives are applicable to every use case or (I)IoT system.

The order of presentation of the functional security primitives roughly correlates to the relative position in the dependency tree. The primitives that are not themselves a dependency to others (in terms of object-oriented programming, they have no parents) are listed first.

A table covering all primitives and their dependencies, as well as the dependency tree, are provided in Table 5 and Figure 3 in the appendix.

4.1 Device Attestation

This functionality provides evidence on the (I)IoT device's (genuine) identity, its software and firmware versions, as well as its integrity and life cycle state. If required, this primitive includes (attested) state indicators of the (I)IoT device and its modules.

NOTE: This primitive applies to the platform, application or both.

Mandatory Dependencies

Device Attestation has the following dependencies:

- (Attested) Secure State and Life Cycle Management: Proof of the (I)IoT device secure state is part of the attestation.
- Cryptographic Operation: The device attestation requires cryptographic functionality, e.g., the computation of a cryptographic hash.
- Genuine Identification: Proof of the genuine, unique identifier of the (I)IoT is provided as part of the attestation.
- Secure Initialization: Device attestation provides evidence on the integrity protection of the system at run-time, which requires a secure initialization of the (I)IoT device.

Optional Dependencies

None

4.2 Secure Updates

This primitive describes the functionality and process to securely update an (I)IoT device in the field. Depending on the device implementation, this might encompass updates and patches of firmware, software, applications, operating system, or a combination thereof, as well as modifying the device configuration and the installation of new applications. Depending on the use case, this may also include downgrades to previous versions in a controlled and secured manner. In that case, the (I)IoT device shall include a mechanism to enforce update policies.

NOTE: This primitive applies to the process, platform, application, or a combination thereof.

Mandatory Dependencies

Secure updates have the following dependencies:

- (Attested) Secure State and Life Cycle Management: Secure updates require secured states as starting and endpoints. Usually, the update itself is performed in a life cycle state with restricted functionality and exposure.
- Cryptographic Operation: Secure updates require cryptographic functionality, e.g., computing a cryptographic hash of the update to be applied.

Optional Dependencies

Secure updates have the following optional dependencies:

- Cryptographic Key and Certificate Store: Secure updates may require the key and certificate store if the OEM uses certificates or public keys to validate the authenticity of updates or decrypt them if required.
- Secure (Encrypted) Storage: Update files or parts thereof may be stored in the secure storage to ensure integrity across power cycles or device reboots.
- Secure Communication (Protocols): A secure update may require secure communication protocols.

4.3 Secure Onboarding and Offboarding

During secure onboarding, the (I)IoT device is connected and bound into a (local) network and, depending on the use case, to the cloud backend. In the IoT domain, this process is usually performed by the end user; for (I)IoT, this is done by the operator.

Offboarding is the reverse process where the device is released from the network. This may be triggered prior to a secure decommissioning or in preparation for resale, in which case a factory reset is potentially performed afterwards.

NOTE: This primitive applies to the process, platform, application, cloud backend, or a combination thereof.

Mandatory Dependencies

Secure onboarding and offboarding have the following dependencies:

- Genuine Identification: Secure onboarding requires a genuine, unique identifier the (I)IoT device uses towards the (local) cloud backend.

Optional Dependencies

Secure onboarding and offboarding have the following optional dependencies:

- (Attested) Secure State and Life Cycle Management: An OEM may choose to use different life cycle states depending on whether the (I)IoT device is onboarded.
- Cryptographic Key and Certificate Store: The onboarding process may require authenticating the device or cloud backend with key material stored in the cryptographic key and certificate store.
- Cryptographic Key Generation and Injection: During onboarding, key material may be generated on the device or injected into it.
- Residual Information Purging: Offboarding (I)IoT device may be accompanied by purging the (I)IoT device.
- Secure Communication (Protocols): The onboarding process usually relies on secure communication protocols.
- Secure Provisioning and Decommissioning: Secure onboarding may leverage OEM/ODM keys.

4.4 Secure Provisioning and Decommissioning

Provisioning of (I)IoT devices is the process of generating and injection (or deriving) key material that an OEM/ODM can trust. This may be done by different technical means, it may be based on the root of trust of the (I)IoT device, and the key material will finally reside on the (I)IoT device. The key material may include public keys or hashes to identify and validate future updates, keys, and certificates to validate the cloud backend identity, secrets for encrypted connections, or device identifiers.

Secure provisioning shall be performed by a trustworthy process that ensures the confidentiality, integrity and authenticity of the OEM/ODM key material. This process may be based on trustworthy environments (often called secure environments), by a secure protocol or by a combination of both. An OEM/ODM may delegate this step to the manufacturer by utilizing pre-provisioned key material established on the platform during a secured (I)IoT platform manufacturing process or based on key material derived thereof.

Decommissioning describes the reverse process, where sensitive data is securely purged once the end-of-life of the (I)IoT device is declared or reached. Performing a factory reset, purging the device for re-sale, or similar actions performed by the end user are covered in secure onboarding and offboarding.

NOTE: This primitive applies to the process, platform, application, or a combination thereof.

Mandatory Dependencies

Secure provisioning and decommissioning have the following dependencies:

- (Attested) Secure State and Life Cycle Management: After provisioning, a life cycle state change is triggered to prevent repeating the provisioning. Similarly, end-of-life is a dedicated secure state that, depending on the use case, may restrict functionality on the device.
- Cryptographic Key Generation and Injection: During provisioning, key material needs to be generated on the device or injected into it.
- Genuine Identification: Secure provisioning requires a genuine, unique identifier of the (I)IoT against which the key material is issued.
- Root of Trust: Secure provisioning assumes trust in the supply chain.

Optional Dependencies

Secure provisioning and decommissioning have the following optional dependencies:

- Cryptographic Key and Certificate Store: The provisioning process may store OEM/ODM key material in the Cryptographic key and certificate store.
- Residual Information Purging: Decommissioning (I)IoT device may be accompanied by purging the (I)IoT device.

4.5 Secure Communication (Protocols)

(I)IoT devices need to communicate securely with each other, cloud backend or a combination thereof. These primitive clusters provide support for secure communication as well related communication protocol support. Examples of such communication could be encrypted buses on a hardware level, but also the GlobalPlatform secure channel protocol or high-level protocols such as hypertext transfer protocol (HTTP) secured with transport layer security (TLS).

NOTE: This primitive applies to the platform, application, cloud backend, or a combination thereof.

Mandatory Dependencies

Secure communication and the related protocols have the following dependencies:

- Cryptographic Operation: Secure communication requires cryptographic functionality such as encryption of the exchanged messages.
- Cryptographic Random Number Generation: Most secure communication protocols require the generation of a random seed or nonce, e.g., for proof of possession of the private key by the communication partner.

Optional Dependencies

Secure communication and the related protocols have the following optional dependencies:

- Cryptographic Key and Certificate Store: The key material stored in the cryptographic key and certificate store may be used for the establishment of a communication session.
- Cryptographic Key Generation and Injection: During the establishment of a communication session, cryptographic keys might be generated.

4.6 Secure Debug and Test

Debugging and testing are essential utilities for developing an (I)IoT device. However, they typically also allow for manipulation of the device state and extracting sensitive data from it. Therefore, they shall be disabled on production devices before being shipped to end users. This primitive encompasses both the controlled disablement of debugging and testing facilities as well as the securing of debugging interfaces.

Both logical debug facilities and physical interfaces need to be protected. Examples for logical debug interfaces contain software APIs dedicated to testing, or debug symbols in compiled code. A physical test interface commonly found in ICs is the Joint Test Action Group (JTAG) interface.

NOTE: This primitive applies to the platform, application or both.

Mandatory Dependencies

Secure debug and test have the following dependencies:

- (Attested) Secure State and Life Cycle Management: Debugging and testing shall only be available in certain life cycle states but not in the field.

Optional Dependencies

Secure debug and test have the following optional dependencies:

- Account Authentication and Management: Some debugging interfaces may require account authentication.

4.7 Secure Backup and Recovery

Secure backup and recovery describes the functionality to back up the (I)IoT device (locally or in the cloud), and may be restored at a later point in time. The backup may include user data, device software, device state, device configuration, or a combination thereof. The backup data shall be integrity and authenticity protected. Backup and recovery may be performed as a part of the device commissioning or onboarding.

Depending on the use case, this functionality may include the functionality to create legitimate clones. However, then it would not be possible to attest a genuine device identification.

NOTE: This primitive applies to the platform, application, cloud backend, or a combination thereof.

Mandatory Dependencies

Secure backup and recovery have the following dependencies:

- Cryptographic Operation: To ensure integrity, authenticity, and, if required, confidentiality of the backup data, cryptographic operations are required.
- Secure (Encrypted) Storage: Backup files or parts thereof shall be stored in the secure storage to ensure the integrity and, if required, confidentiality.

Optional Dependencies

Secure backup and recovery have the following optional dependencies:

- (Attested) Secure State and Life Cycle Management: Backups may only be available in certain life cycle states.
- Cryptographic Key and Certificate Store: Secure backup and recovery may require key material stored in the key and certificate store.

4.8 Account Authentication and Management

This primitive collects functionality to identify and authenticate the user and (I)IoT device accounts. User accounts are typically owned by end users who have signed-up for the OEM's (I)IoT system. Device accounts may be used in scenarios of private clouds or in dedicated industrial networks, where machines identify themselves to the cloud backend without user interaction. This primitive includes the process of managing such accounts and encompasses processes and technical means for on- and offboarding of accounts, suspending and resuming accounts and similar functionality. It may include authorization and access control management.

NOTE: This primitive applies to processes, application, cloud backend, or a combination thereof.

Mandatory Dependencies

Account authentication and management have the following dependencies:

- Cryptographic Operation: The account authentication requires cryptographic functionality.
- Secure (Encrypted) Storage: User credentials need to be stored in a secured manner.

Optional Dependencies

Account authentication and management have the following optional dependencies:

- Cryptographic Key and Certificate Store: Account credentials may be stored in the cryptographic key and certificate store.

4.9 (Attested) Secure State and Life Cycle Management

This primitive and its related implementation ensures that an (I)IoT device is in a defined, secured life cycle state. Optionally, this primitive also encompasses functionality to provide evidence on the device state. If required, secure life cycle transitions of the device and policies for such transition, as well as proof of the correctness of transition, may be part of the life cycle management.

In this work, no life cycle states are explicitly defined. However, a few dedicated states are assumed to be present to enable security primitives that depend on life cycle management:

- A "manufacturing life cycle state" that allows the commissioning of the device, including the generation or injection of key material; development of the (I)IoT device with debugging and testing facilities may be enabled in this state or a dedicated one
- An "in-field life cycle state" with disabled debugging and testing facilities intended for end user (I)IoT devices during normal operation.
- A "decommissioned life cycle state" that prohibits onboarding of the (I)IoT devices to the (local) cloud.

A good starting point on life cycles and their transitions in the context of secure (I)IoT devices is given in GlobalPlatform[2].

NOTE: This primitive applies to the platform, application, or both.

Mandatory Dependencies

(Attested) Secure state and life cycle management have the following dependencies:

- Anomaly Detection and Reaction: Mutual dependency—maintaining a secure state requires proper detection and reaction of anomalies.
- Secure Initialization: A secure state can only be reached through secure initialization.

Optional Dependencies

None

4.10 Genuine Identification

Genuine identification is the functionality to emit a unique identification of an (I)IoT device. The identification may be realized as a unique identifier, such as a serial number stored on the platform or may be derived from platform features. Optionally, this identification is physically unclonable and is used as part of fraud prevention and detection.

Proof of this identity is not covered here but is part of the device attestation.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

None

Optional Dependencies

Genuine identification has the following optional dependencies:

- Cryptographic Operation: Genuine identification may be cryptographically computed.

4.11 Secure Initialization

This primitive ensures the authenticity and integrity of the device bootloader, firmware, and other software during the boot process and ensures that the intended secure life cycle state is reached. If required, the implementation may handle confidentiality protected (encrypted) boot code.

Depending on the use case, secure initialization may encompass one or more boot stages that are each cryptographically secured. Secure initialization may also include validating and securely starting of the application running on the platform.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

Secure initialization has the following dependencies:

- Cryptographic Operation: Secure Initialization requires cryptographic functionality; at the very minimum, the computation of a cryptographic hash of the boot image.

Optional Dependencies

Secure initialization has the following optional dependencies:

- Cryptographic Key and Certificate Store: Secure initialization may require key material stored in the Cryptographic key and certificate store.

4.12 Anomaly Detection and Reaction

This primitive describes the process or algorithm that analyzes the (I)IoT device input and output, such as sensor data, as well as the software integrity and application operation for abnormal events and, if required, triggers and executes an action. Typically these actions encompass logging the anomaly, issuing a message to the cloud backend, resetting the device, and/or changing a secure life cycle state. Especially in safety-critical domains, a detected anomaly would trigger transitioning into a fail-safe state of operation.

This primitive includes logical and physical tamper detection (stand-alone or as an input to the detection algorithm) and tamper protection. Monitoring of the cloud backend also falls into this category. It may also cover error handling, e.g., in case of software anomalies.

NOTE: This primitive applies to the platform, application, backend, or a combination thereof.

Mandatory Dependencies

Anomaly detection and reaction have the following dependencies:

- (Attested) Secure State and Life Cycle Management: Mutual dependency—upon detection of an anomaly, a secure life cycle state change shall be triggered if the operation of the (I)IoT device is compromised. This may either be realized as a transition into a fail-safe or error state or by performing a power-cycle on the (I)IoT device followed by a secure initialization in order to re-establish a secure state. In some cases where a reaction may severely impact the functional operation or safety, it may be required to mark the (I)IoT system state compromised instead of transitioning to another (I)IoT device state.

Optional Dependencies

Anomaly detection and reaction have the following optional dependencies:

- System Event Logging: Upon detection of an anomaly, a system event may be logged securely.
- Secure (Encrypted) Storage: Upon detection of an anomaly, the secure storage may be wiped.
- Residual Information Purging: Upon detection of an anomaly, the device RAM may be wiped.

4.13 Cryptographic Key Generation and Injection

This item describes functionality to securely generate cryptographic keys and optionally to securely inject or import them into the (I)IoT device. The implementation may support key exchange and key agreement support, as well as key derivation schemes. If a cryptographic key and certificate store is present, an interface shall be provided to generate or store the keys in the secure key store.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

Cryptographic key generation and injection have the following dependencies:

- Cryptographic Operation: Cryptographic key generation and injection requires cryptographic functionality.
- Cryptographic Random Number Generation: Cryptographic key generation and injection requires the generation of (true) random numbers.

Optional Dependencies

Cryptographic key generation and injection has the following optional dependencies:

- Cryptographic Key and Certificate Store: If present, key generation and injection shall leverage the cryptographic key and certificate store.

4.14 Cryptographic Key and Certificate Store

The cryptographic key and certificate store allows the user to store key material such as keys and certificates and enforce policies on them. The key and certificate store shall provide (non-cryptographic) management functionality for the key material such as policy management or key material deletion.

If the use case requires a key export, the cryptographic key and certificate store shall provide policy management to mark key material as non-exportable and enforce this policy by technical means.

The policy management may provide additional flags for key material such as limitations on usage.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

The cryptographic key and certificate store have the following dependencies:

- (Attested) Secure State and Life Cycle Management: Operations on the key and certificate store shall only be available in the (I)IoT device is in a secure life cycle state.
- Cryptographic Operation: The cryptographic key and certificate store provides cryptographic functionality on the key material it holds.
- Secure (Encrypted) Storage: Key material is stored in the secure encrypted storage.

Optional Dependencies

The cryptographic key and certificate Store has the following optional dependencies:

- Residual Information Purging: If the underlying platform supports it, the cryptographic key and certificate store shall purge the memory regions used for its operations.
- Software Isolation: If the underlying platform supports it, operations of the cryptographic key and certificate store shall be executed in isolation.

4.15 Secure (Encrypted) Storage

Secure storage provides functionality to store data securely and maintain its integrity. If required, it may provide additional functionality such as encryption to protect data confidentiality.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

The secure (encrypted) storage has the following dependencies:

- Cryptographic Operation: The secure storage requires cryptographic functionality to provide integrity protection and, if required, ensure the confidentiality of the stored data.

Optional Dependencies

None

4.16 Cryptographic Operation

This primitive groups cryptographic functionality such as encryption, decryption, hashing, or signing. Depending on the platform and use case, these might be provided by a dedicated secure element, by specific hardware features, or by a cryptographic library or software stack used by the application. In the latter case, the security framework, libraries, or software stack provided by the platform shall be used. If a cryptographic key and certificate store is present, an interface shall be provided to leverage this functionality utilizing the keys in the secure key store.

If required by the use case, cryptographic operation may include higher-level functionality such as certificate verification, certificate signing, and certificate signing request (CSR) handling.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

None

Optional Dependencies

Cryptographic operation has the following optional dependencies:

- Software Isolation: If the underlying platform supports it, cryptographic operations shall be executed in isolation.

4.17 Cryptographic Random Number Generation

For many secure protocols and related cryptographic functionality, it is required to generate random numbers securely. Optionally, this primitive includes the generation of true random numbers.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

None

Optional Dependencies

None

4.18 System Event Logging

Most (I)IoT devices require facilities to (securely) log system events in an integrity-protected way.

This primitive may be used to implement means of ensuring non-repudiation.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

System event logging has the following dependencies:

- Secure (Encrypted) Storage: Events and related data are stored in the secure encrypted storage.

Optional Dependencies

System event logging has the following optional dependencies:

- Monotonic Time: System event logging may use monotonic counters or timestamps to ensure integrity on the order of events.

4.19 Root of Trust

This primitive relates to the initial root of trust (RoT) on the security component that is established during the manufacturing process and is the foundation for the device commissioning. This might be achieved, for instance, by manufacturing the (I)IoT device inside trusted manufacturing facilities, or, if available, by using pre-provisioned secure elements in a zero-trust environment.

NOTE: This primitive applies to the process, platform, or both.

Mandatory Dependencies

None

Optional Dependencies

The root of trust has the following optional dependencies:

- Cryptographic Operation: The root of trust may use cryptographic functionality to derive device identity or key material.

4.20 Residual Information Purging

This functionality ensures that deallocated data is no longer present; for instance, that a newly allocated and not yet initialized memory does not contain (parts of) its previous content. This covers data in volatile memory and optionally non-volatile memories.

One implementation that falls into this primitives is the blanking of cryptographic keys.

NOTE: This primitive applies to the platform, application or both.

Mandatory Dependencies

None

Optional Dependencies

None

4.21 Software Isolation

This primitive describes means to isolate the device operating system (OS) from applications, as well as applications from each other. This includes the separation of resources such as memory regions claimed by the OS, applications or a combination of both.

This may be realized by moving secure applications, cryptographic functionality, or both into a dedicated secure subsystem or secure element.

NOTE: This primitive applies to the platform.

Mandatory Dependencies

None

Optional Dependencies

None

4.22 Monotonic Time

Rollback and replay protection, as well as mechanisms for non-repudiation, require monotonically increasing counters or timestamps or similar. This primitive includes measures in hardware or software (for instance, leveraging blockchains) to provide measures of monotonically increasing time.

NOTE: This primitive applies to the platform, application, or both.

Mandatory Dependencies

None

Optional Dependencies

None

5 Definition of Security Process Primitives

This section covers process-related security primitives of (I)IoT systems. As depicted in Figure 1, processes typically encompass the whole development process and operation of the IoT device. As such, these primitives are applicable to the platform, the application and the cloud backend, as well as the process category itself.

5.1 Secure Policy Compliance

This primitive describes compliance of the (I)IoT device functionality, as well as related development and operational processes to local and global security policies and legislation.

Mandatory Dependencies

Secure policy compliance has the following dependencies:

- Security by Design: Most secure policies and processes require security to be considered during the design phase.
- Vulnerability and Incident Management: Most policies and processes mandate a vulnerability and incident management process.
- Protection of Personal Information: Most regulations mandate the protection of personal information.

Optional Dependencies

None

5.2 Security by Design

This primitive describes a process to ensure security best practices are followed during the (I)IoT device development and manufacturing phase. It also mandates baseline security for the device configuration and (end user) credentials.

Mandatory Dependencies

None

Optional Dependencies

Security by design has the following optional dependencies:

- Account Authentication and Management: Security by design mandates policies on account management if such functionality is available.

5.3 Vulnerability and Incident Management

Processes to allow third parties to report flaws and vulnerabilities and react on them, as well as to disclose vulnerabilities and incidents to end users and authorities. This is mandated by many regulations, such as the European General Data Protection Regulation (GDPR).

Mandatory Dependencies

None

Optional Dependencies

None

5.4 Protection of Personal Information

Protection of personally identifiable information of end users and compliance with corresponding legislation such as GDPR.

Mandatory Dependencies

Protection of personal information has the following dependencies:

- Cryptographic Operation: Cryptographic functionality is required to ensure the confidentiality of personal information.
- Secure (Encrypted) Storage: Personal data shall be stored in secure encrypted storage.

Optional Dependencies

Protection of personal information has the following optional dependencies:

- Cryptographic Key and Certificate Store: If present, end user key material shall be stored in the key and certificate store.

6 Conclusion

In this document, a nomenclature in the form of security primitives is presented for IoT security requirements. The security primitives are defined, consisting of meaningful and non-overlapping categories of security features and requirements. A procedure is described for how the primitives can be aligned with common standards, and finally, how the resulting set of primitives can be mapped to a particular product for the use case. Thus, based on this analysis, one has a structured set of information to continue to evaluate the product's functional sufficiency, its security requirements. To illustrate the procedure, one particular use case and the example of the ISA/IEC 62443 standard is shown how the security primitives can be mapped to this specific use case and to the detailed requirements of the standard. This yields the security functional requirements an (I)IoT device needs to fulfill, and as such, helps to identify the respective product features and primitives needed to meet the requirements of the standard.

After defining the terminology and approach and showing its applicability to a specific use case and standard, the recommended next steps are to prove the concept along further use cases and standards. Following this, security levels can be defined based on such a commonly agreed terminology and discussed to show the robustness of a particular implementation.

Appendix A Full Mapping Table

An electronic version of the mapping table is provided on request.

Appendix B Auxiliary Material

B.1 Security Primitive Dependency Table

The full relation of the security primitives detailed in Sections 4 and 5 is compiled into the format of a table and given in Table 5. In this table, all dependencies of a security primitive are given in a row, with direct mandatory dependencies depicted as a filled circle. By recursively considering the mandatory dependencies, all indirect dependent primitives are identified and shown as an empty circle. Direct optional dependencies are given as an empty square. Please refer to the respective section of the security primitive for a rationale on the dependencies.

	Device Attestation	Secure Updates	Secure Onboarding and Offboarding	Secure Provisioning and Decommissioning	Secure Communication (Protocols)	Secure Debug and Test	Secure Backup and Recovery	Account Authentication and Management	(Attested) Secure State and Life Cycle Management	Genuine Identification	Secure Initialization	Anomaly Detection and Reaction	Cryptographic Key Generation and Injection	Cryptographic Key and Certificate Store	Secure (Encrypted) Storage	Cryptographic Operation	Cryptographic Random Number Generation	System Event Logging	Root of Trust	Residual Information Purging	Monotonic Time	Secure Policy Compliance	Security by Design	Software Isolation	Vulnerability and Incident Management	Protection of Personal Information
Device Attestation									●	●	●					●										
Secure Updates					□				●		○			□	□	●										
Secure Onboarding and Offboarding				□	□				□	●			□	□												
Secure Provisioning and Decommissioning									●	●	○		●	□		○	○		●							
Secure Communication (Protocols)													□	□		●	●									
Secure Debug and Test								□	●		○					○										
Secure Backup and Recovery									□					□	●	●										
Account Authentication and Management														□	●	●										
(Attested) Secure State and Life Cycle Management											●					○										
Genuine Identification																□										
Secure Initialization														□		●										
Anomaly Detection and Reaction									●		○				□	○		□		□						
Cryptographic Key Generation and Injection														□		●	●									
Cryptographic Key and Certificate Store									●		○				●	●				□				□		
Secure (Encrypted) Storage																●										
Cryptographic Operation																										
Cryptographic Random Number Generation																										
System Event Logging															●	○					□					
Root of Trust																□										
Residual Information Purging																										
Software Isolation																										
Monotonic Time																										
Secure Policy Compliance									○		○			○	○	○						●			●	●
Security by Design								□																		
Vulnerability and Incident Management																										
Protection of Personal Information									○		○			●	●	●										

Table 5: Dependency Table of the Security Primitives.
● indicates mandatory dependencies, ○ denotes indirect mandatory dependencies and □ is for optional dependencies.

B.2 Security Primitive Dependency Graph

The complete dependency graph resulting from the dependencies outlined in Sections 4 and 5 is given in Figure 3. Solid lines denote mandatory requirements, while dashed lines refer to optional ones. Please refer to the respective section of the parent for a rationale on the dependency.

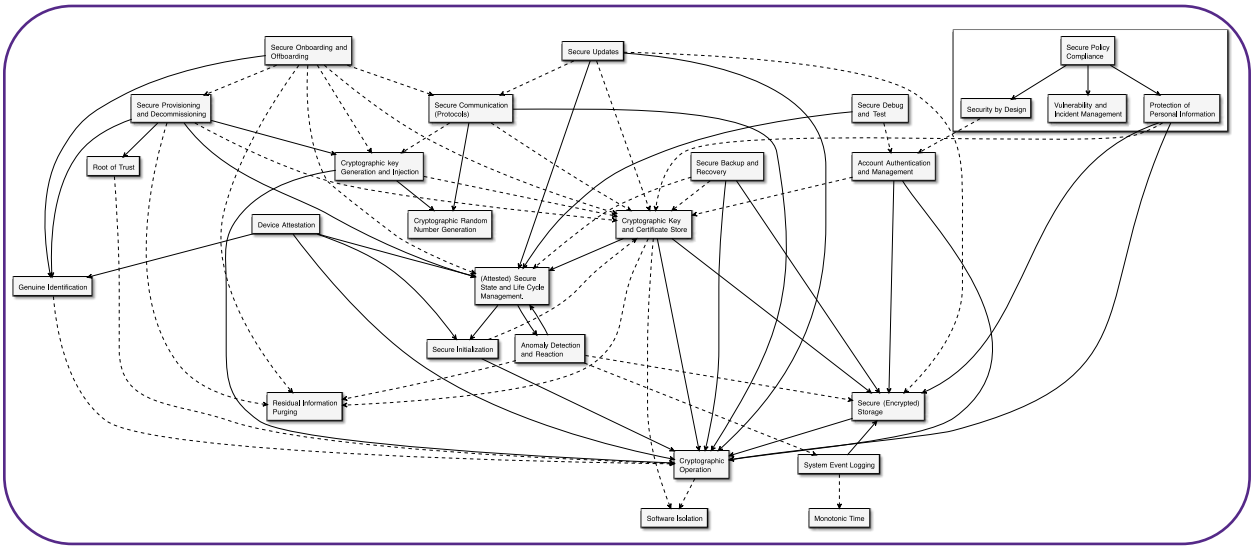

Figure 3: Dependency Graph of the Security Primitives. — denote mandatory dependencies, -------- are optional dependencies.

B.3 Glossary

A glossary of the abbreviations used in this document is given in Table 6.

Abbreviation	Description
GDPR	General Data Protection Regulation
ECC	Elliptic-curve cryptography
HTTP	Hypertext transfer protocol
(I)IoT	(Industrial) Internet of Things
IT	Information technology
JTAG	Joint Test Action Group
MCU	Microcontroller unit
MPU	Microprocessor unit
OS	Operating system
OEMs/ODMs	Original Equipment Manufacturer/Original Device Manufacturer
PP	Protection profile
PSA	Platform security architecture
RAM	Random access memory
RoT	Root of trust
RSA	Rivest–Shamir–Adleman (Cryptosystem)
SESIP	Security evaluation standard for IoT platforms
SFR	Security functional requirement
SoC	System on chip
SPU	Secure processing unit
TEE	Trusted execution environment
TLS	Transport layer security

Table 6: Table of Abbreviations

B.4 References

[1] GlobalPlatform, "IoTopia: A comprehensive framework for IoT security," [Online]. Available: https://globalplatform.org/iotopia/. [Accessed 20 January 2020].

[2] GlobalPlatform, Security Evaluation Standard for IoT Platforms (SESIP), Version 0.0.0.5, 2019.

[3] U.K. Government, "Secure by Design," 28 February 2019. [Online]. Available: https://www.gov.uk/government/collections/secure-by-design. [Accessed 22 November 2019].

[4] U.S. Congress, Internet of Things Cybersecurity Improvement Act of 2019, 2019.

[5] M. Fagan, K. N. Megas, K. Scarfone and M. Smith, NISTIR 8259: Recommendations for IoT Device Manufacturers: Foundational Activities and Core Device Cybersecurity Capability Baseline (2nd Draft), Gaithersburg, MD: National Institute of Standards and Technology, 2020.

[6] International Society of Automation, "New ISA/IEC 62443 standard specifies security capabilities for control system components," [Online]. Available: https://www.isa.org/intech/201810standards/. [Accessed 20 January 2020].

[7] TRAFICOM Finnish Transport and Communications Agency National Cyber Security Centre, "Finland becomes the first European country to certify safe smart devices – new Cybersecurity label helps consumers buy safer products," 26 November 2019. [Online]. Available: https://www.kyberturvallisuuskeskus.fi/en/news/finland-becomes-first-european-country-certify-safe-smart-devices-new-cybersecurity-label. [Accessed 10 December 2019].

[8] ETSI, "ETSI EN 303 645: Cyber Security for Consumer Internet of Things, Draft V2.0.0," November 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.00.00_20/en_303645v020000a.pdf. [Accessed 28 February 2020].

[9] SAE International, "SAE J3101: Hardware Protected Security for Ground Vehicles," 10 February 2020. [Online]. Available: https://www.sae.org/standards/content/j3101_202002/. [Accessed 14 January 2020].

[10] National Institute of Standards and Technology, FIPS PUB 140-2: Security Requirements for Cryptographic Modules, Gaithersburg, MD, 2001.

[11] National Institute of Standards and Technology, FIPS PUB 140-3: Security Requirements for Cryptographic Modules, Gaithersburg, MD, 2019.

[12] GlobalPlatform Device Committee, TEEProtection Profile, version 1.2, 2014.

[13] Arm Ltd., "Platform Security Architecture," [Online]. Available: https://www.arm.com/why-arm/architecture/platform-security-architecture. [Accessed 2 March 2020].

[14] STMicroelectronics NV , "AN5156: Introduction to STM32 microcontrollers security, rev. 4," 2020.

[15] S. Bradner, RFC2119: Key words for use in RFCs to Indicate Requirement Levels, 1997.

[16] NXP Semiconductors, i.MX RT1050 Crossover MCUs for Industrial Products, rev. 1.4, Eindhoven, Netherlands, 2020.

[17] NXP Semiconductors, SE050 Plug & Trust Secure Element - Objective Data Sheet, Eindhoven, Netherlands, 2019.

[18] NXP Semiconductors, AN12400: SE050 for secure connection to OEM cloud, Eindhoven, Netherlands, 2019.

[19] NXP Semiconductors, AN12278: LPC55S69 Security Solutions for IoT, Eindhoven, Netherlands, 2019.

How to Reach Us:

Home Page: **www.nxp.com**
Web Support: **www.nxp.com/support**

USA/Europe or Locations Not Listed:
NXP Semiconductors USA, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.nxp.com/support

Europe, Middle East, and Africa:
NXP Semiconductors Germany GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.nxp.com/support

Japan:
NXP Japan Ltd.
Yebisu Garden Place Tower 24F,
4-20-3, Ebisu, Shibuya-ku,
Tokyo 150-6024, Japan
0120 950 032 (Domestic Toll Free)
https://www.nxp.jp/
https://www.nxp.com/support/support:SUPPORTHOME

Asia/Pacific:
NXP Semiconductors Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@nxp.com

Document Number: GREENBOX2WPA4 REV 0