NIST SPECIAL PUBLICATION 1800-36B

Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management:

Enhancing Internet Protocol-Based IoT Device and Network Security

Volume B:
Approach, Architecture, and Security Characteristics

Michael Fagan
Jeffrey Marron
Paul Watrobski
Murugiah Souppaya
National Cybersecurity Center of Excellence
Information Technology Laboratory

William Barker
Dakota Consulting
Silver Spring, Maryland

Chelsea Deane
Joshua Klosterman
Charlie Rearick
Blaine Mulugeta
Susan Symington
The MITRE Corporation
McLean, Virginia

Dan Harkins
Danny Jump
Aruba, a Hewlett Packard Enterprise Company
San Jose, California

Andy Dolan
Kyle Haefner
Craig Pratt
Darshak Thakore
CableLabs
Louisville, Colorado

Peter Romness
Cisco
San Jose, California

Tyler Baker
David Griego
Foundries.io
London, United Kingdom

Brecht Wyseur
Kudelski IoT
Cheseaux-sur-Lausanne, Switzerland

Alexandru Mereacre
Nick Allott
Ashley Setter
NquiringMinds
Southampton, United Kingdom

Julien Delplancke
NXP Semiconductors
Mougins, France

Michael Richardson
Sandelman Software Works
Ontario, Canada

Steve Clark
SEALSQ, a subsidiary of WISeKey
Geneva, Switzerland

Mike Dow
Steve Egerter
Silicon Labs
Austin, Texas

May 2024

DRAFT

This publication is available free of charge from
https://www.nccoe.nist.gov/projects/trusted-iot-device-network-layer-onboarding-and-lifecycle-management

DISCLAIMER

Certain commercial entities, equipment, products, or materials may be identified by name or company logo or other insignia in order to acknowledge their participation in this collaboration or to describe an experimental procedure or concept adequately. Such identification is not intended to imply special status or relationship with NIST or recommendation or endorsement by NIST or NCCoE; neither is it intended to imply that the entities, equipment, products, or materials are necessarily the best available for the purpose.

National Institute of Standards and Technology Special Publication 1800-36B, Natl. Inst. Stand. Technol. Spec. Publ. 1800-36B, 114 pages, May 2024, CODEN: NSPUE2

FEEDBACK

You can improve this guide by contributing feedback regarding which aspects of it you find helpful as well as suggestions on how it might be improved. Should we provide guidance summaries that target specific audiences? What trusted IoT device onboarding protocols and related features are most important to you? Is there some content that is not included in this document that we should cover? Are we missing anything in terms of technologies or use cases? In what areas would it be most helpful for us to focus our future related efforts? For example, should we consider implementing builds that onboard devices supporting Matter and/or the Fast Identity Online (FIDO) Alliance application onboarding protocol? Should we implement builds that integrate security mechanisms such as lifecycle management, supply chain management, attestation, or behavioral analysis? As you review and adopt this solution for your own organization, we ask you and your colleagues to share your experience and advice with us.

Comments on this publication may be submitted to: iot-onboarding@nist.gov.

Public comment period: May 31, 2024 through July 30, 2024

All comments are subject to release under the Freedom of Information Act.

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, MD 20899
Email: nccoe@nist.gov

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

The National Cybersecurity Center of Excellence (NCCoE), a part of the National Institute of Standards and Technology (NIST), is a collaborative hub where industry organizations, government agencies, and academic institutions work together to address businesses' most pressing cybersecurity issues. This public-private partnership enables the creation of practical cybersecurity solutions for specific industries, as well as for broad, cross-sector technology challenges. Through consortia under Cooperative Research and Development Agreements (CRADAs), including technology partners—from Fortune 50 market leaders to smaller companies specializing in information technology security—the NCCoE applies standards and best practices to develop modular, adaptable example cybersecurity solutions using commercially available technology. The NCCoE documents these example solutions in the NIST Special Publication 1800 series, which maps capabilities to the NIST Cybersecurity Framework and details the steps needed for another entity to re-create the example solution. The NCCoE was established in 2012 by NIST in partnership with the State of Maryland and Montgomery County, Maryland.

To learn more about the NCCoE, visit https://www.nccoe.nist.gov/. To learn more about NIST, visit https://www.nist.gov.

NIST CYBERSECURITY PRACTICE GUIDES

NIST Cybersecurity Practice Guides (Special Publication 1800 series) target specific cybersecurity challenges in the public and private sectors. They are practical, user-friendly guides that facilitate the adoption of standards-based approaches to cybersecurity. They show members of the information security community how to implement example solutions that help them align with relevant standards and best practices, and provide users with the materials lists, configuration files, and other information they need to implement a similar approach.

The documents in this series describe example implementations of cybersecurity practices that businesses and other organizations may voluntarily adopt. These documents do not describe regulations or mandatory practices, nor do they carry statutory authority.

KEYWORDS

application-layer onboarding; bootstrapping; Internet of Things (IoT); Manufacturer Usage Description (MUD); network-layer onboarding; onboarding; Wi-Fi Easy Connect.

ACKNOWLEDGMENTS

We are grateful to the following individuals for their generous contributions of expertise and time.

Name	Organization
Amogh Guruprasad Deshmukh	Aruba, a Hewlett Packard Enterprise company
Bart Brinkman	Cisco

Name	Organization
Eliot Lear	Cisco
George Grey	Foundries.io
David Griego	Foundries.io
Fabien Gremaud	Kudelski IoT
Faith Ryan	The MITRE Corporation
Toby Ealden	NquiringMinds
John Manslow	NquiringMinds
Antony McCaigue	NquiringMinds
Alexandru Mereacre	NquiringMinds
Loic Cavaille	NXP Semiconductors
Mihai Chelalau	NXP Semiconductors
Julien Delplancke	NXP Semiconductors
Anda-Alexandra Dorneanu	NXP Semiconductors
Todd Nuzum	NXP Semiconductors
Nicusor Penisoara	NXP Semiconductors
Laurentiu Tudor	NXP Semiconductors
Karen Scarfone	Scarfone Cybersecurity
Pedro Fuentes	SEALSQ, a subsidiary of WISeKey
Gweltas Radenac	SEALSQ, a subsidiary of WISeKey
Kalvin Yang	SEALSQ, a subsidiary of WISeKey

The Technology Partners/Collaborators who participated in this build submitted their capabilities in response to a notice in the Federal Register. Respondents with relevant capabilities or product components were invited to sign a Cooperative Research and Development Agreement (CRADA) with NIST, allowing them to participate in a consortium to build this example solution. We worked with:

Technology Collaborators		
Aruba, a Hewlett Packard Enterprise company	Foundries.io	Open Connectivity Foundation (OCF)
CableLabs	Kudelski IoT	Sandelman Software Works
Cisco	NquiringMinds	SEALSQ, a subsidiary of WISeKey
	NXP Semiconductors	Silicon Labs

DOCUMENT CONVENTIONS

The terms “shall” and “shall not” indicate requirements to be followed strictly to conform to the publication and from which no deviation is permitted. The terms “should” and “should not” indicate that among several possibilities, one is recommended as particularly suitable without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited. The terms “may” and “need not” indicate a course of action permissible within the limits of the publication. The terms “can” and “cannot” indicate a possibility and capability, whether material, physical, or causal.

CALL FOR PATENT CLAIMS

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

1. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
2. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: iot-onboarding@nist.gov.

Contents

List of Figures

List of Tables

1 Summary

IoT devices are typically connected to a network. As with any other device needing to communicate on a network securely, an IoT device needs credentials that are specific to that network to help ensure that only authorized devices can connect to and use the network. A typical commercially available, mass-produced IoT device cannot be pre-provisioned with local network credentials by the manufacturer during the manufacturing process. Instead, the local network credentials will be provisioned to the device at the time of its deployment. This practice guide is focused on trusted methods of providing IoT devices with the network-layer credentials and policy they need to join a network upon deployment, a process known as *network-layer onboarding.*

Establishing trust between a network and an IoT device (as defined in NIST Internal Report 8425) prior to providing the device with the credentials it needs to join the network is crucial for mitigating the risk of potential attacks. There are two possibilities for attack. One is where a device is convinced to join an unauthorized network, which would take control of the device. The other is where a network is infiltrated by a malicious device. Trust is achieved by attesting and verifying the identity and posture of the device and the network before providing the device with its network credentials—a process known as *network-layer onboarding*. In addition, scalable, automated mechanisms are needed to safely manage IoT devices throughout their lifecycles, such as safeguards that verify the security posture of a device before the device is permitted to execute certain operations.

In this practice guide, the National Cybersecurity Center of Excellence (NCCoE) applies standards, best practices, and commercially available technology to demonstrate various mechanisms for trusted network-layer onboarding of IoT devices. This guide shows how to provide network credentials to IoT devices in a trusted manner and maintain a secure device posture throughout the device lifecycle.

1.1 Challenge

With 40 billion IoT devices expected to be connected worldwide by 2025 [1], it is unrealistic to onboard or manage these devices by visiting each device and performing a manual action. While it is possible for devices to be securely provided with their local network credentials at the time of manufacture, this requires the manufacturer to customize network-layer onboarding on a build-to-order basis, which prevents the manufacturer from taking full advantage of the economies of scale that could result from building identical devices for all its customers.

The industry lacks scalable, automatic mechanisms to safely manage IoT devices throughout their lifecycles and lacks a trusted mechanism for providing IoT devices with their network credentials and policy at the time of deployment on the network. It is easy for a network to falsely identify itself, yet many IoT devices onboard to networks without verifying the network's identity and ensuring that it is their intended target network. Also, many IoT devices lack user interfaces, making it cumbersome to manually input network credentials. Wi-Fi is sometimes used to provide credentials over an open (i.e., unencrypted) network, but this onboarding method risks credential disclosure. Most home networks use a single password shared among all devices, so access is controlled only by the device's possession of the password and does not consider a unique device identity or whether the device belongs on the network. This method also increases the risk of exposing credentials to unauthorized parties. Providing unique credentials to each device is more secure, but doing so manually would be resource-intensive and error-prone, would risk credential disclosure, and cannot be performed at scale.

Once a device is connected to the network, if it becomes compromised, it can pose a security risk to both the network and other connected devices. Not keeping such a device current with the most recent software and firmware updates may make it more susceptible to compromise. The device could also be attacked through the receipt of malicious payloads. Once compromised, it may be used to attack other devices on the network.

1.2 Solution

We need scalable, automated, trusted mechanisms to safely manage IoT devices throughout their lifecycles to ensure that they remain secure, starting with secure ways to provision devices with their network credentials, i.e., beginning with network-layer onboarding. Onboarding is a particularly vulnerable point in the device lifecycle because if it is not performed in a secure manner, then both the device and the network are at risk. Networks are at risk of having unauthorized devices connect to them, and devices are at risk of being taken over by networks that are not authorized to onboard or control them.

The NCCoE has adopted the trusted network-layer onboarding approach to promote automated, trusted ways to provide IoT devices with unique network credentials and manage devices throughout their lifecycles to ensure that they remain secure. The NCCoE is collaborating with CRADA consortium technology providers in a phased approach to develop example implementations of trusted network-layer onboarding solutions. We define a *trusted network-layer onboarding solution* to be a mechanism for provisioning network credentials to a device that:

- provides each device with unique network credentials,
- enables the device and the network to mutually authenticate,
- sends devices their network credentials over an encrypted channel,
- does not provide any person with access to the network credentials, and
- can be performed repeatedly throughout the device lifecycle to enable:
  - the device’s network credentials to be securely managed and replaced as needed, and
  - the device to be securely onboarded to other networks after being repurposed or resold.

The use cases designed to be demonstrated by this project’s implementations include:

- trusted network-layer onboarding of IoT devices
- repeated trusted network-layer onboarding of devices to the same or a different network
- automatic establishment of an encrypted connection between an IoT device and a trusted application service (i.e., *trusted application-layer onboarding*) after the IoT device has performed trusted network-layer onboarding and used its credentials to connect to the network
- policy-based ongoing device authorization
- software-based methods to provision device birth credentials in the factory

- mechanisms for IoT device manufacturers to provide IoT device purchasers with information needed to onboard the IoT devices to their networks (i.e., *device bootstrapping information*)

1.3 Benefits

This practice guide can benefit both IoT device users and IoT device manufacturers. The guide can help IoT device users understand how to onboard IoT devices to their networks in a trusted manner to:

- Ensure that their network is not put at risk as IoT devices are added to it
- Safeguard their IoT devices from being taken over by unauthorized networks
- Provide IoT devices with unique credentials for network access
- Provide, renew, and replace device network credentials in a secure manner
- Ensure that IoT devices can automatically and securely perform application-layer onboarding after performing trusted network-layer onboarding and connecting to a network
- Support ongoing protection of IoT devices throughout their lifecycles

This guide can help IoT device manufacturers, as well as manufacturers and vendors of semiconductors, secure storage components, and network onboarding equipment, understand the desired security properties for supporting trusted network-layer onboarding and demonstrate mechanisms for:

- Placing unique credentials into secure storage on IoT devices at time of manufacture (i.e., *device birth credentials*)
- Installing onboarding software onto IoT devices
- Providing IoT device purchasers with information needed to onboard the IoT devices to their networks (i.e., *device bootstrapping information*)
- Integrating support for network-layer onboarding with additional security capabilities to provide ongoing protection throughout the device lifecycle

2 How to Use This Guide

This NIST Cybersecurity Practice Guide demonstrates a standards-based reference design for implementing trusted IoT device network-layer onboarding and lifecycle management and describes various example implementations of this reference design. Each of these implementations, which are known as *builds,* is standards-based and is designed to help provide assurance that networks are not put at risk as new IoT devices are added to them and help safeguard IoT devices from connecting to unauthorized networks. The reference design described in this practice guide is modular and can be deployed in whole or in part, enabling organizations to incorporate trusted IoT device network-layer onboarding and lifecycle management into their legacy environments according to goals that they have prioritized based on risk, cost, and resources.

NIST is adopting an agile process to publish this content. Each volume is being made available as soon as possible rather than delaying release until all volumes are completed.

This guide contains five volumes:

- NIST Special Publication (SP) 1800-36A: *Executive Summary* – why we wrote this guide, the challenge we address, why it could be important to your organization, and our approach to solving this challenge
- NIST SP 1800-36B: *Approach, Architecture, and Security Characteristics* – what we built and why **(you are here)**
- NIST SP 1800-36C: *How-To Guides* – instructions for building the example implementations, including all the security-relevant details that would allow you to replicate all or parts of this project
- NIST SP 1800-36D: *Functional Demonstrations* – use cases that have been defined to showcase trusted IoT device network-layer onboarding and lifecycle management security capabilities, and the results of demonstrating these use cases with each of the example implementations
- NIST SP 1800-36E: *Risk and Compliance Management* – risk analysis and mapping of trusted IoT device network-layer onboarding and lifecycle management security characteristics to cybersecurity standards and recommended practices

Depending on your role in your organization, you might use this guide in different ways:

Business decision makers, including chief security and technology officers, will be interested in the *Executive Summary, NIST SP 1800-36A*, which describes the following topics:

- challenges that enterprises face in migrating to the use of trusted IoT device network-layer onboarding
- example solutions built at the NCCoE
- benefits of adopting the example solution

Technology or security program managers who are concerned with how to identify, understand, assess, and mitigate risk will be interested in *NIST SP 1800-36B*, which describes what we did and why.

Also, Section 4 of *NIST SP 1800-36E* will be of particular interest. Section 4, *Mappings*, maps logical components of the general trusted IoT device network-layer onboarding and lifecycle management reference design to security characteristics listed in various cybersecurity standards and recommended practices documents, including *Framework for Improving Critical Infrastructure Cybersecurity* (NIST Cybersecurity Framework) and *Security and Privacy Controls for Information Systems and Organizations* (NIST SP 800-53).

You might share the *Executive Summary, NIST SP 1800-36A*, with your leadership team members to help them understand the importance of using standards-based implementations for trusted IoT device network-layer onboarding and lifecycle management.

IT professionals who want to implement similar solutions will find all volumes of the practice guide useful. You can use the how-to portion of the guide, *NIST SP 1800-36C*, to replicate all or parts of the builds created in our lab. The how-to portion of the guide provides specific product installation, configuration, and integration instructions for implementing the example solution. We do not re-create the product manufacturers' documentation, which is generally widely available. Rather, we show how we incorporated the products together in our environment to create an example solution. Also, you can use *Functional Demonstrations, NIST SP 1800-36D*, which provides the use cases that have been defined to showcase trusted IoT device network-layer onboarding and lifecycle management security capabilities and the results of demonstrating these use cases with each of the example implementations. Finally, *NIST SP 1800-36E* will be helpful in explaining the security functionality that the components of each build provide.

This guide assumes that IT professionals have experience implementing security products within the enterprise. While we have used a suite of commercial products to address this challenge, this guide does not endorse these particular products. Your organization can adopt this solution or one that adheres to these guidelines in whole, or you can use this guide as a starting point for tailoring and implementing parts of a trusted IoT device network-layer onboarding and lifecycle management solution. Your organization's security experts should identify the products that will best integrate with your existing tools and IT system infrastructure. We hope that you will seek products that are congruent with applicable standards and recommended practices.

A NIST Cybersecurity Practice Guide does not describe "the" solution, but example solutions. We seek feedback on the publication's contents and welcome your input. Comments, suggestions, and success stories will improve subsequent versions of this guide. Please contribute your thoughts to iot-onboarding@nist.gov.

2.1 Typographic Conventions

The following table presents typographic conventions used in this volume.

Typeface/Symbol	Meaning	Example
Italics	file names and path names; references to documents that are not hyperlinks; new terms; and placeholders	For language use and style guidance, see the *NCCoE Style Guide*.
Bold	names of menus, options, command buttons, and fields	Choose **File** > **Edit**.
`Monospace`	command-line input, onscreen computer output, sample code examples, and status codes	`mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's NCCoE are available at https://www.nccoe.nist.gov.

3 Approach

This project builds on the document-based research presented in the NIST Draft Cybersecurity White Paper, *Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management* [2]. That paper describes key security and other characteristics of a trusted network-layer onboarding solution as well as the integration of onboarding with related technologies such as device attestation, device communications intent [3][4], and application-layer onboarding. The security and other attributes of the onboarding process that are cataloged and defined in that paper can provide assurance that the network is not put at risk as new IoT devices are added to it and also that IoT devices are safeguarded from being taken over by unauthorized networks.

To kick off this project, the NCCoE published a Federal Register Notice [5] inviting technology providers to participate in demonstrating approaches to deploying trusted IoT device network-layer onboarding and lifecycle management in home and enterprise networks, with the objective of showing how trusted IoT device network-layer onboarding can practically and effectively enhance the overall security of IoT devices and, by extension, the security of the networks to which they connect. The Federal Register Notice invited technology providers to provide products and/or expertise to compose prototypes. Components sought included network onboarding components and IoT devices that support trusted network-layer onboarding protocols; authorization services; supply chain integration services; access points, routers, or switches; components that support device communications intent management; attestation services; controllers or application services; IoT device lifecycle management services; and asset management services. Cooperative Research and Development Agreements (CRADAs) were established with qualified respondents, and teams of collaborators were assembled to build a variety of implementations.

NIST is following an agile methodology of building implementations iteratively and incrementally, starting with network-layer onboarding and gradually integrating additional capabilities that improve device and network security throughout a managed device lifecycle. The project team began by designing a general, protocol-agnostic reference architecture for trusted network-layer onboarding (see Section 4) and establishing a laboratory infrastructure at the NCCoE to host implementations (see Section 5).

Five build teams were established to implement trusted network-layer onboarding prototypes, and a sixth build team was established to demonstrate multiple builds for factory provisioning activities performed by an IoT device manufacturer to enable devices to support trusted network-layer onboarding. Each of the build teams fleshed out the initial architectures of their example implementations. They then used technologies, capabilities, and components from project collaborators to begin creating the builds:

- Build 1 (Wi-Fi Easy Connect, Aruba/HPE) uses components from Aruba, a Hewlett Packard Enterprise company, to support trusted network-layer onboarding using the Wi-Fi Alliance's Wi-Fi Easy Connect Specification, Version 2.0 [6] and independent (see Section 3.3.2) application-layer onboarding to the Aruba User Experience Insight (UXI) cloud.
- Build 2 (Wi-Fi Easy Connect, CableLabs, OCF) uses components from CableLabs to support trusted network-layer onboarding using the Wi-Fi Easy Connect protocol that allows provisioning of per-device credentials and policy management for each device. Build 2 also uses components from the Open Connectivity Foundation (OCF) to support streamlined (see Section 3.3.2) trusted application-layer onboarding to the OCF security domain.
- Build 3 (BRSKI, Sandelman Software Works) uses components from Sandelman Software Works to support trusted network-layer onboarding using the Bootstrapping Remote Secure Key Infrastructure (BRSKI) [7] protocol and an independent, third-party Manufacturer Authorized Signing Authority (MASA).

- Build 4 (Thread [8], Silicon Labs, Kudelski IoT) uses components from Silicon Labs to support connection to an OpenThread [9] network using pre-shared credentials and components from Kudelski IoT to support trusted application-layer onboarding to the Amazon Web Services (AWS) IoT core.
- Build 5 (BRSKI over Wi-Fi, NquiringMinds) uses components from NquiringMinds to support trusted network-layer onboarding using the BRSKI protocol over 802.11 [10]. Additional components from NquiringMinds support ongoing, policy-based, continuous assurance and authorization, as well as device communications intent enforcement.
- The BRSKI Factory Provisioning Build uses components from NquiringMinds to implement the factory provisioning flows. The build is implemented on Raspberry Pi devices, where the IoT secure element is an integrated Infineon Optiga™ SLB 9670 TPM 2.0. The device certificate authority (CA) is externally hosted on NquiringMinds servers. This build demonstrates activities for provisioning IoT devices with their initial (i.e., birth—see Section 3.3) credentials for use with the BRSKI protocol and for making device bootstrapping information available to device owners.
- The Wi-Fi Easy Connect Factory Provisioning Build uses Raspberry Pi devices and code from Aruba and secure storage elements, code, and a CA from SEALSQ, a subsidiary of WISeKey. This build demonstrates activities for provisioning IoT devices with their birth credentials for use with the Wi-Fi Easy Connect protocol and for making device bootstrapping information available to device owners.

Each build team documented the architecture and design of its build (see Appendix C, Appendix D, Appendix E, Appendix F, Appendix G, and Appendix H). As each build progressed, its team also documented the steps taken to install and configure each component of the build (see NIST SP 1800-36C).

The project team then designed a set of use case scenarios designed to showcase the builds' security capabilities. Each build team conducted a functional demonstration of its build by running the build through the defined scenarios and documenting the results (see NIST SP 1800-36D).

The project team also conducted a risk assessment and a security characteristic analysis and documented the results, including mappings of the security capabilities of the reference solution to both the *Framework for Improving Critical Infrastructure Cybersecurity* (NIST Cybersecurity Framework) [11] and Security and Privacy Controls for Information Systems and Organizations (*NIST SP 800-53 Rev. 5*) (see NIST SP 1800-36E).

Finally, the NCCoE worked with industry and standards-developing organization collaborators to distill their findings and consider potential enhancements to future support for trusted IoT device network-layer onboarding (see Section 6 and Section 7).

3.1 Audience

The intended audience for this practice guide includes:

- IoT device manufacturers, integrators, and vendors
- Semiconductor manufacturers and vendors
- Secure storage manufacturers

- Network equipment manufacturers
- IoT device owners and users
- Owners and administrators of networks (both home and enterprise) to which IoT devices connect
- Service providers (internet service providers/cable operators and application platform providers)

3.2 Scope

This project focuses on the trusted network-layer onboarding of IoT devices in both home and enterprise environments. Enterprise, consumer, and industrial use cases for trusted IoT device network-layer onboarding are all considered to be in scope at this time. The project encompasses trusted network-layer onboarding of IoT devices deployed across different Internet Protocol (IP) based environments using wired, Wi-Fi, and broadband networking technologies. The project addresses the onboarding of IP-based devices in the initial phase and will consider using technologies such as Zigbee or Bluetooth in future phases of this project.

The project's scope also includes security technologies that can be integrated with and enhanced by the trusted network-layer onboarding mechanism to protect the device and its network throughout the device's lifecycle. Examples of these technologies include supply chain management, device attestation, trusted application-layer onboarding, device communications intent enforcement, device lifecycle management, asset management, the dynamic assignment of devices to various network segments, and ongoing device authorization. Aspects of these technologies that are relevant to their integration with network-layer onboarding are within scope. Demonstration of the general capabilities of these technologies independent of onboarding is not within the project's scope. For example, demonstrating a policy that requires device attestation to be performed before the device will be permitted to be onboarded would be within scope. However, the details and general operation of the device attestation mechanism would be out of scope.

3.3 Assumptions and Definitions

This project is guided by a variety of assumptions, which are categorized by subsection below.

3.3.1 Credential Types

There are several different credentials that may be related to any given IoT device, which makes it important to be clear about which credential is being referred to. Two types of IoT device credentials are involved in the network-layer onboarding process: birth credentials and network credentials. Birth credentials are installed onto the device before it is released into the supply chain; trusted network-layer onboarding solutions leverage birth credentials to authenticate devices and securely provision them with their network credentials. If supported by the device and the application service provider, application-layer credentials may be provisioned to the device after the device performs network-layer onboarding and connects to the network, during the application-layer onboarding process. These different types of IoT device credentials are defined as follows:

- **Birth Credential**: In order to participate in trusted network-layer onboarding, devices must be equipped with a birth credential, which is sometimes also referred to as a device *birth identity* or *birth certificate.* A birth credential is a unique, authoritative credential that is generated or installed into secure storage on the IoT device during the pre-market phase of the device's lifecycle, i.e., before the device is released for sale. A manufacturer, integrator, or vendor typically generates or installs the birth credential onto an IoT device in the form of an Initial Device Identifier (IDevID) [12] and/or a public/private key pair.

  Birth credentials:

  - are permanent, and their value is independent of context;
  - enable the trusted network-layer onboarding process while keeping the device manufacturing process efficient; and
  - include a unique identity and a secret and can range from simple raw public and private keys to X.509 certificates that are signed by a trusted authority.

- **Network Credential:** A network credential is the credential that is provisioned to an IoT device during network-layer onboarding. The network credential enables the device to connect to the local network securely. A device's network credential may be changed repeatedly, as needed, by subsequent invocation of the trusted network-layer onboarding process.

Additional types of credentials that may also be associated with an IoT device are:

- **Application-Layer Credential:** An application-layer credential is a credential that is provisioned to an IoT device during application-layer onboarding. After an IoT device has performed network-layer onboarding and connected to a network, it may be provisioned with one or more application-layer credentials during the application-layer onboarding process. Each application-layer credential is specific to a given application and is typically unique to the device, and it may be replaced repeatedly over the course of the device's lifetime.
- **User Credential:** An IoT device that permits authorized users to access it and restricts access only to authorized users will have one or more user credentials associated with it. These credentials are what the users present to the IoT device in order to gain access to it. The user credential is not relevant during network-layer onboarding and is generally not of interest within the scope of this project. We include it in this list only for completeness. Many IoT devices may not even have user credentials associated with them.

In order to perform network- and application-layer onboarding, the device being onboarded must already have been provisioned with birth credentials. A pre-provisioned, unique, authoritative birth credential is essential for enabling the IoT device to be identified and authenticated as part of the trusted network-layer onboarding process, no matter what network the device is being onboarded to or how many times it is onboarded. The value of the birth credential is independent of context, whereas the network credential that is provisioned during network-layer onboarding is significant only with respect to the network to which the IoT device will connect. Each application-layer credential that is provisioned during application-layer onboarding is specific to a given application, and each user credential is specific to a given user. A given IoT device only ever has one birth credential over the course of its lifetime, and the value of this birth credential remains unchanged. However, that IoT device may have any number of network, application-layer, and user credentials at any given point in time, and these credentials may be replaced repeatedly over the course of the device's lifetime.

3.3.2 Integrating Security Enhancements

Integrating trusted network-layer IoT device onboarding with additional security mechanisms and technologies can help increase trust in both the IoT device and the network to which it connects. Examples of such security mechanism integrations demonstrated in this project include:

- **Trusted Application-Layer Onboarding:** When supported, application-layer onboarding can be performed automatically after a device has connected to its local network. Trusted application-layer onboarding enables a device to be securely provisioned with the application-layer credentials it needs to establish a secure association with a trusted application service. In many cases, a network's IoT devices will be so numerous that manually onboarding devices at the application layer would not be practical; in addition, dependence on manual application-layer onboarding would leave the devices vulnerable to accidental or malicious misconfiguration. So, application-layer onboarding, like network-layer onboarding, is fundamental to ensuring the overall security posture of each IoT device.

  As part of the application-layer onboarding process, devices and the application services with which they interact perform mutual authentication and establish an encrypted channel over which the application service can download application-layer credentials and software to the device and the device can provide information to the application service, as appropriate. Application-layer onboarding is useful for ensuring that IoT devices are executing the most up-to-date versions of their intended applications. It can also be used to establish a secure association between a device and a trusted lifecycle management service, which will ensure that the IoT device continues to be patched and updated with the latest firmware and software, thereby enabling the device to remain trusted throughout its lifecycle.

  Network-layer onboarding cannot be performed until after network-layer bootstrapping information has been introduced to the device and the network. This network-layer bootstrapping information enables the device and the network to mutually authenticate and establish a secure channel. Analogously, application-layer onboarding cannot be performed until after application-layer bootstrapping information has been introduced to the device and the application servers with which they will onboard. This application-layer bootstrapping information enables the device and the application server to mutually authenticate and establish a secure channel.

  - *Streamlined Application-Layer Onboarding*—One potential mechanism for introducing this application-layer bootstrapping information to the device and the application server is to use the network-layer onboarding process. The secure channel that is established during network-layer onboarding can serve as the mechanism for exchanging application-layer bootstrapping information between the device and the application server. By safeguarding the integrity and confidentiality of the application-layer bootstrapping information as it is conveyed between the device and the application server, the trusted network-layer onboarding mechanism helps to ensure that information that the device and the application server use to authenticate each other is truly secret and known only to them, thereby establishing a firm foundation for their secure association. In this way, trusted network-layer onboarding can provide a secure foundation for trusted application-layer onboarding. We call an application-layer onboarding process that uses network-layer onboarding to exchange application-layer bootstrapping information *streamlined* application-layer onboarding.

   - *Independent Application-Layer Onboarding*—An alternative mechanism for introducing application-layer bootstrapping information to the device is to provide this information to the device during the manufacturing process. During manufacturing, the IoT device can be provisioned with software and associated bootstrapping information that enables the device to mutually authenticate with an application-layer service after it has connected to the network. This mechanism for performing application-layer onboarding does not rely on the network-layer onboarding process to provide application-layer bootstrapping information to the device. All that is required is that the device have connectivity to the application-layer onboarding service after it has connected to the network. We call an application-layer onboarding process that does not rely on network-layer onboarding to exchange application-layer bootstrapping information *independent* application-layer onboarding.

- **Segmentation:** Upon connection to the network, a device may be assigned to a particular local network segment to prevent it from communicating with other network components, as determined by enterprise policy. The device can be protected from other local network components that meet or do not meet certain policy criteria. Similarly, other local network components may be protected from the device if it meets or fails to meet certain policy criteria. A trusted network-layer onboarding mechanism may be used to convey information about the device that can be used to determine to which network segment it should be assigned upon connection. By conveying this information in a manner that protects its integrity and confidentiality, the trusted network-layer onboarding mechanism helps to increase assurance that the device will be assigned to the appropriate network segment. Post-onboarding, if a device becomes untrustworthy, for example because it is found to have software that has a known vulnerability or misconfiguration, or because it is behaving in a suspicious manner, the device may be dynamically assigned to a different network segment as a means of quarantining it, or its network-layer credential can be revoked or deleted.

- **Ongoing Device Authorization:** Once a device has been network-layer onboarded in a trusted manner and has possibly performed application-layer onboarding as well, it is important that as the device continues to operate on the network, it maintains a secure posture throughout its lifecycle. Ensuring the ongoing security of the device is important for keeping the device from being corrupted and for protecting the network from a potentially harmful device. Even though a device is authenticated and authorized prior to being onboarded, it is recommended that the device be subject to ongoing policy-based authentication and authorization as it continues to operate on the network. This may include monitoring device behavior and constraining communications to and from the device as needed in accordance with policy. In this manner, an ongoing device authorization service can ensure that the device and its operations continue to be authorized throughout the device's tenure on the network.

- **Device Communications Intent Enforcement:** Network-layer onboarding protocols can be used to securely transmit device communications intent information from the device to the network (i.e., to transmit this information in encrypted form with integrity protections). After the device has securely connected to the network, the network can use this device communications intent information to ensure that the device sends and receives traffic only from authorized locations. Secure conveyance of device communications intent information, combined with enforcement of it, ensures that IoT devices are constrained to sending and receiving only those communications that are explicitly required for each device to fulfill its purpose.

- **Additional Security Mechanisms:** Although not demonstrated in the implementations that have been built in this project so far, numerous additional security mechanisms can potentially be integrated with network-layer onboarding, beginning at device boot-up and extending through all phases of the device lifecycle. Examples of such mechanisms include integration with supply chain management tools, device attestation, automated lifecycle management, mutual attestation, and centralized asset management. Overall, application of these and other security protections can create a dependency chain of protections. This chain is based on a hardware root of trust as its foundation and extends up to support the security of the trusted network-layer onboarding process. The trusted network-layer onboarding process in turn may enable additional capabilities and provide a foundation that makes them more secure, thereby helping to ensure the ongoing security of the device and, by extension, the network.

3.3.3 Device Limitations

The security capabilities that any onboarding solution will be able to support will depend in part on the hardware, processing power, cryptographic modules, secure storage capacity, battery life, human interface (if any), and other capabilities of the IoT devices themselves, such as whether they support verification of firmware at boot time, attestation, application-layer onboarding, and device communications intent enforcement; what onboarding and other protocols they support; and whether they are supported by supply-chain tools. The more capable the device, the more security capabilities it should be able to support and the more robustly it should be able to support them. Depending on both device and onboarding solution capabilities, different levels of assurance may be provided.

3.3.4 Specifications Are Still Improving

Ideally, trusted network-layer onboarding solutions selected for widespread implementation and use will be openly available and standards-based. Some potential solution specifications are still being improved. In the meantime, their instability may be a limiting factor in deploying operational implementations of the proposed capabilities. For example, the details of running BRSKI over Wi-Fi are not fully specified at this time.

3.4 Collaborators and Their Contributions

Organizations participating in this project submitted their capabilities in response to an open call in the Federal Register for all sources of relevant security capabilities from academia and industry (vendors and integrators). Listed below are the respondents with relevant capabilities or product components (identified as “Technology Partners/Collaborators” herein) who signed a CRADA to collaborate with NIST in a consortium to build example trusted IoT device network-layer onboarding solution.

Technology Collaborators		
Aruba, a Hewlett Packard Enterprise company	Foundries.io	Open Connectivity Foundation (OCF)
CableLabs	Kudelski IoT	Sandelman Software Works
Cisco	NquiringMinds	SEALSQ, a subsidiary of WISeKey
	NXP Semiconductors	Silicon Labs

Table 3-1 summarizes the capabilities and components provided, or planned to be provided, by each partner/collaborator.

Table 3-1 Capabilities and Components Provided by Each Technology Partner/Collaborator

Collaborator	Security Capability or Component Provided
Aruba	Infrastructure for trusted network-layer onboarding using the Wi-Fi Easy Connect protocol and application-layer onboarding to the UXI cloud. IoT devices for use with both Wi-Fi Easy Connect network-layer onboarding and application-layer onboarding. The UXI Dashboard provides for an “always-on” remote technician with near real-time data insights into network and application performance.
CableLabs	Infrastructure for trusted network-layer onboarding using the Wi-Fi Easy Connect protocol. IoT devices for use with both Wi-Fi Easy Connect network-layer onboarding and application-layer onboarding to the OCF security domain.
Cisco	Networking components to support various builds.
Foundries.io	Factory software for providing birth credentials into secure storage on IoT devices and for transferring device bootstrapping information from device manufacturer to device purchaser.
Kudelski IoT	Infrastructure for trusted application-layer onboarding of a device to the AWS IoT core. The service comes with a cloud platform and a software agent that enables secure provisioning of AWS credentials into the secure storage of IoT devices.
NquiringMinds	Infrastructure for trusted network-layer onboarding using BRSKI over 802.11. Service that performs ongoing monitoring of connected devices to ensure their continued authorization (i.e., continuous authorization service), as well as device communications intent enforcement.
NXP Semiconductors	IoT devices with secure storage for use with both Wi-Fi Easy Connect and BRSKI network-layer onboarding. Service for provisioning credentials into secure storage of IoT devices.
Open Connectivity Foundation (OCF)	Infrastructure for trusted application-layer onboarding to the OCF security domain using IoTivity, an open-source software framework that implements the OCF specification.
Sandelman Software Works	Infrastructure for trusted network-layer onboarding using BRSKI. IoT devices for use with BRSKI network-layer onboarding.
SEALSQ, a subsidiary of WISeKey	Secure storage elements, code, and software that simulates factory provisioning of birth credentials to those secure elements on IoT devices in support of both Wi-Fi Easy Connect and BRSKI network-layer onboarding; certificate authority for signing device certificates.
Silicon Labs	Infrastructure for connection to a Thread network that has access to other networks for application-layer onboarding. IoT device with secure storage for use with Thread network connection and application-layer onboarding using Kudelski IoT.

Each of these technology partners and collaborators has described the relevant products and capabilities it brings to this trusted onboarding effort in the following subsections. The NCCoE does not certify or validate products or services. We demonstrate the capabilities that can be achieved by using participants' contributed technology.

3.4.1 Aruba, a Hewlett Packard Enterprise Company

Aruba, a Hewlett Packard Enterprise (HPE) company, provides secure, intelligent edge-to-cloud networking solutions that use artificial intelligence (AI) to automate the network, while harnessing data to drive powerful business outcomes. With Aruba ESP (Edge Services Platform) and as-a-service options as part of the HPE GreenLake family, Aruba takes a cloud-native approach to helping customers meet their connectivity, security, and financial requirements across campus, branch, data center, and remote worker environments, covering all aspects of wired, wireless local area networking (LAN), and wide area networking (WAN). Aruba ESP provides unified solutions for connectivity, visibility, and control throughout the IT-IoT workflow, with the objective of helping organizations accelerate IoT-driven digital transformation with greater ease, efficiency, and security. To learn more, visit Aruba at https://www.arubanetworks.com/.

3.4.1.1 Device Provisioning Protocol

Device Provisioning Protocol (DPP), certified under the Wi-Fi Alliance (WFA) as "Easy Connect," is a standard developed by Aruba that allows IoT devices to be easily provisioned onto a secure network. DPP improves security by leveraging Wi-Fi Protected Access 3 (WPA3) to provide device-specific credentials, enhance certificate handling, and support robust, secure, and scalable provisioning of IoT devices in any commercial, industrial, government, or consumer application. Aruba implements DPP through a combination of on-premises hardware and cloud-based services as shown in Table 3-1.

3.4.1.2 Aruba Access Point (AP)

From their unique vantage as ceiling furniture, Aruba Wi-Fi 6 APs have an unobstructed overhead view of all nearby devices. Built-in Bluetooth Low Energy (BLE) and Zigbee 802.15.4 IoT radios, as well as a flexible USB port, provide IoT device connectivity that allows organizations to address a broad range of IoT applications with infrastructure already in place, eliminating the cost of gateways and IoT overlay networks while enhancing IoT security.

Aruba's APs enable a DPP network through an existing Service Set Identifier (SSID) enforcing DPP access control and advertising the Configurator Connectivity Information Element (IE) to attract unprovisioned clients (i.e., clients that have not yet been onboarded). Paired with Aruba's cloud management service "Central", the APs implement the DPP protocol. The AP performs the DPP network introduction protocol (Connector exchange) with provisioned clients and assigns network roles.

3.4.1.3 Aruba Central

Aruba Central is a cloud-based networking solution with AI-powered insights, workflow automation, and edge-to-cloud security that empowers IT teams to manage and optimize campus, branch, remote, data center, and IoT networks from a single point of visibility and control. Built on a cloud-native, microservices architecture, Aruba Central is designed to simplify IT and IoT operations, improve agility, and reduce costs by unifying management of all network infrastructure.

Aruba's "Central" Cloud DPP service exposes and controls many centralized functions to enable a seamless integrated end-to-end solution and act as a DPP service orchestrator. The cloud based DPP service selects an AP to authenticate unprovisioned enrollees (in the event that multiple APs receive the client *chirps*). The DPP cloud service holds the Configurator signing key and generates Connectors for enrollees authenticated through an AP.

3.4.1.4 IoT Operations

Available within Aruba Central, the IoT Operations service extends network administrators' view into IoT devices and applications connected to the network. Organizations can gain critical visibility into previously invisible IoT devices, as well as reduce costs and complexity associated with deploying IoT applications. IoT Operations comprises three core elements:

- IoT Dashboard, which provides a granular view of devices connected to Aruba APs, as well as IoT connectors and applications in use.
- IoT App Store, a repository of click-and-go IoT applications that interface with IoT devices and their data.
- IoT Connector, which provisions multiple applications to be computed at the edge for agile IoT application support.

3.4.1.5 Client Insights

Part of Aruba Central, AI-powered Client Insights automatically identifies each endpoint connecting to the network with up to 99% accuracy. Client Insights discovers and classifies all connected endpoints—including IoT devices—using built-in machine learning and dynamic profiling techniques, helping organizations better understand what's on their networks, automate access privileges, and monitor the behavior of each endpoint's traffic flows to more rapidly spot attacks and act.

3.4.1.6 Cloud Auth

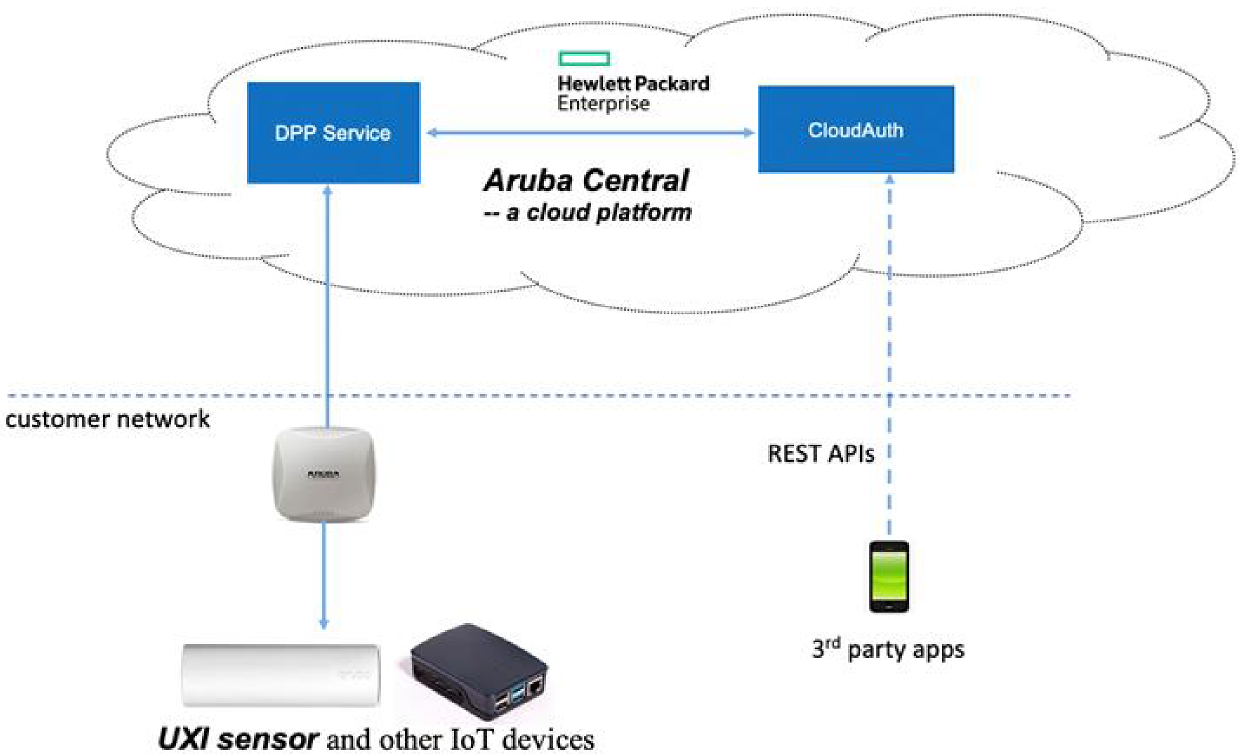
Cloud-native network access control (NAC) solution Cloud Auth delivers time-saving workflows to configure and manage onboarding, authorization, and authentication policies for wired and wireless networks. Cloud Auth integrates with an organization's existing cloud identity store, such as Google Workspace or Azure Active Directory, to authenticate IoT device information and assign the right level of network access.

Cloud Auth operates as the DPP Authorization server and is the repository for trusted DPP Uniform Resource Identifiers (URIs) of unprovisioned enrollees. It maintains role information for each unprovisioned DPP URI and provisioned devices based on unique per-device credential (public key extracted from Connector). Representational State Transfer (RESTful) application programming interfaces (APIs) provide extensible capabilities to support third parties, making an easy path for integration and collaborative deployments.

3.4.1.7 UXI Sensor: DPP Enrollee

User Experience Insight (UXI) sensors continuously monitor end-user experience on customer networks and provide a simple-to-use cloud-based dashboard to assess networks and applications. The UXI sensor is onboarded in a zero-touch experience using DPP. Once network-layer onboarding is complete, the UXI sensor performs application-layer onboarding to the Aruba cloud to download a customer-specific profile. This profile enables the UXI sensor to perform continuous network testing and monitoring, and to troubleshoot network issues that it finds.

Figure 3-1 Aruba/HPE DPP Onboarding Components



3.4.2 CableLabs

CableLabs is an innovation lab for future-forward research and development (R&D)—a global meeting of minds dedicated to building and orchestrating emergent technologies. By convening peers and experts to share knowledge, CableLabs' objective is to energize the industry ecosystem for speed and scale. Its research facilitates solutions with the goal of making connectivity faster, easier, and more secure, and its conferences and events offer neutral meeting points to gain consensus.

As part of this project, CableLabs has provided the reference platform for its Custom Connectivity architecture for the purpose of demonstrating trusted network-layer onboarding of Wi-Fi devices using a variety of credentials. The following components are part of the reference platform.

3.4.2.1 Platform Controller

The controller provides interfaces and messaging for managing service deployment groups, access points with the deployment groups, registration and lifecycle of user services, and the secure onboarding and lifecycle management of users' Wi-Fi devices. The controller also exposes APIs for integration with third-party systems for the purpose of integrating various business flows (e.g., integration with manufacturing process for device management).

3.4.2.2 Custom Connectivity Gateway Agent

The Gateway Agent is a software component that resides on the Wi-Fi AP and gateway. It connects with the controller to coordinate the Wi-Fi and routing capabilities on the gateway. Specifically, it enforces the policies and configuration from the controller by managing the lifecycle of the Wi-Fi Extended Service Set/Basic Service Set (ESS/BSS) on the AP, authentication and credentials of the client devices that connect to the AP, and service management and routing rules for various devices. It also manages secure onboarding capabilities like Easy Connect, simple onboarding using a per-device pre-shared key (PSK), etc. The Gateway agent is provided in the form of an operational Raspberry Pi-based Gateway that also includes hostapd for Wi-Fi/DPP and open-vswitch for the creation of trust domains and routing.

3.4.2.3 Reference Clients

Three Raspberry Pi-based reference clients are provided. The reference clients have support for WFA Easy Connect-based onboarding as well as support for different Wi-Fi credentials, including per-device PSK and 802.1x certificates. One of the reference clients also has support for OCF-based streamlined application-layer onboarding.

3.4.3 Cisco

Cisco Systems, or Cisco, delivers collaboration, enterprise, and industrial networking and security solutions. The company's cybersecurity team, Cisco Secure, is one of the largest cloud and network security providers in the world. Cisco's Talos Intelligence Group, the largest commercial threat intelligence team in the world, is comprised of world-class threat researchers, analysts, and engineers, and supported by unrivaled telemetry and sophisticated systems. The group feeds rapid and actionable threat intelligence to Cisco customers, products, and services to help identify new threats quickly and defend against them. Cisco solutions are built to work together and integrate into your environment, using the "network as a sensor" and "network as an enforcer" approach to both make your team more efficient and keep your enterprise secure. Learn more about Cisco at https://www.cisco.com/go/secure.

3.4.3.1 Cisco Catalyst Switch

A Cisco Catalyst switch is provided to support network connectivity and network segmentation capabilities.

3.4.4 Foundries.io

Foundries.io helps organizations bring secure IoT and edge devices to market faster. The FoundriesFactory cloud platform offers DevOps teams a secure Linux-based firmware/operating system (OS) platform with device and fleet management services for connected devices, based on a fixed no-royalty subscription model. Product development teams gain enhanced security from boot to cloud while reducing the cost of developing, deploying, and updating devices across their installed lifetime. The open-source platform interfaces to any cloud and offers Foundries.io customers maximum flexibility for hardware configuration, so organizations can focus on their intellectual property, applications, and value add. For more information, please visit https://foundries.io/.

3.4.4.1 FoundriesFactory

FoundriesFactory is a cloud-based software platform provided by Foundries.io that offers a complete development and deployment environment for creating secure IoT devices. It provides a set of tools and services that enable developers to create, test, and deploy custom firmware images, as well as manage the lifecycle of their IoT devices.

Customizable components include open-source secure boot software, the open-source Linux microPlatform (LmP) distribution built with Yocto and designed for secure managed IoT and edge products, secure Over the Air (OTA) update facilities, and a Docker runtime for managing containerized applications and services. The platform is cross architecture (x86, Arm, and RISC-V) and enables secure connections to public and private cloud services.

Leveraging open standards and open software, FoundriesFactory is designed to simplify and accelerate the process of developing, deploying, and managing IoT and edge devices at scale, while also ensuring that they are secure and up to date over the product lifetime.

3.4.5 Kudelski IoT

Kudelski IoT is the Internet of Things division of Kudelski Group and provides end-to-end IoT solutions, IoT product design, and full-lifecycle services to IoT semiconductor and device manufacturers, ecosystem creators, and end-user companies. These solutions and services leverage the group's 30+ years of innovation in digital business model creation; hardware, software, and ecosystem design and testing; state-of-the-art security lifecycle management technologies and services; and managed operation of complex systems.

3.4.5.1 Kudelski IoT keySTREAM™

Kudelski IoT keySTREAM is a device-to-cloud, end-to-end solution for securing all the key assets of an IoT ecosystem during its entire lifecycle. The system provides each device with a unique, immutable, unclonable identity that forms the foundation for critical IoT security functions like in-factory or in-field provisioning, data encryption, authentication, and secure firmware updates, as well as allowing companies to revoke network access for vulnerable devices if necessary. This ensures that the entire lifecycle of the device and its data can be managed.

In this project, keySTREAM is used to enable trusted application-layer onboarding. It manages the attestation of devices, ownership, and provisioning of application credentials.

3.4.6 NquiringMinds

NquiringMinds provides intelligent trusted systems, combining AI-powered analytics with cyber security fundamentals. tdx Volt is the NquiringMinds general-purpose zero-trust services infrastructure platform, upon which it has built Cyber tdx, a cognitively enhanced cyber defense service designed for IoT. Both products are the latest iteration of the TDX product family. NquiringMinds is a UK company. Since 2010, it has been deploying its solutions into smart cities, health care, industrial, agricultural, financial technology, defense, and security sectors.

NquiringMinds collaborates within the open-standards and open-source community. It focuses on the principle of continuous assurance: the ability to continually reassess security risk by intelligently reasoning across the hard and soft information sources available. NquiringMinds' primary contributions to this project, described in the subsections below, are being made available as open source.

3.4.6.1 NquiringMinds' BRSKI Protocol Implementation

NquiringMinds has open sourced their software implementation of IETF's Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol, which provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) devices. This implementation includes the necessary adaptations for BRKSI to work with Wi-Fi networks.

The open source BRSKI implementation is available under an Apache 2.0 license at: https://github.com/nqminds/brski

3.4.6.2 TrustNetZ

NquiringMinds has open sourced the TrustNetZ (Zero Trust Networking) software stack which sits on top of their BRSKI implementation. TrustNetZ embodies the network onboarding and lifecycle management concepts into an easy to replicate demonstrator which includes the IoT device, the router, the router onboarding, the registrar, the manufacturer, the manufacturer provisioning, policy enforcement and continuous assurance servers.

This software also encapsulates NquiringMinds' continuous assurance capability, enhancing the security of the network by continually assessing whether connected IoT devices meet the policy requirements of the network. The software also includes a flexible, verifiable credential-based policy framework, which can rapidly be adapted to model different security and business model scenarios. The implementation models networking onboarding flows with EAP-TLS Wi-Fi certificates.

The open source TrustNetZ implementation is available under an Apache 2.0 license at: https://github.com/nqminds/trustnetz

3.4.6.3 edgeSEC

edgeSEC is an open-source, OpenWrt-based implementation of an intelligent secure router. It implements, on an open stack, the key components needed to implement both trusted onboarding and continuous assurance of devices. It contains an implementation of the Internet Engineering Task Force (IETF) BRSKI protocols, with the necessary adaptations for wireless onboarding, fully integrated into an open operational router. It additionally implements device communications intent constraints (IETF Manufacturer Usage Description [MUD]) and behavior monitoring (IoTSF ManySecured) that support some of the more enhanced trusted onboarding use cases. EdgeSEC additionally provides the platform for an asynchronous control plane for the continuous management of multiple routers and a general-purpose policy evaluation point, which can be used to demonstrate the breadth of onboarding and monitoring use cases that can be supported.

EdgeSEC is not directly used in the build that was demonstrated for this project, but it contains critical pieces of code that have been adapted in a simplified manner for the TrustNetZ implementation.

The open source edgeSEC implementation is available under an Apache 2.0 license at: https://github.com/nqminds/edgesec

3.4.6.4 tdx Volt

tdx Volt is NquiringMinds' zero-trust infrastructure platform. It encapsulates identity management, credential management, service discovery, and smart policy evaluation. This platform is designed to simplify the end-to-end demonstration of the trusted onboarding process and provides tools for use on the IoT device, the router, applications, and clouds. Tdx Volt is used by the TrustNetZ demonstrator as a verifiable credential issuer and verifier.

Tdx Volt is an NquiringMinds' product. Documented working implementation are available at: https://docs.tdxvolt.com/en/introduction

3.4.6.5 Reference Hardware

For demonstration purposes the NquiringMinds components can be deployed using the following hardware:

Compute hosts: Raspberry Pi 4

https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. The Raspberry Pis are used to host the IoT client device, the router, and all additional compute services. Other Raspberry Pi models are also likely to work but have not been tested.

TPM/Secure Element

The secure storage for the IoT device (used in network-layer onboarding and factory provisioning) is provided by an Infineon Optiga™ SLB 9670 TPM 2.0, integrated through a Geeek Pi TPM hat. https://www.infineon.com/dgdl/Infineon-OPTIGA_SLx_9670_TPM_2.0_Pi_4-ApplicationNotes-v07_19-EN.pdf?fileId=5546d4626c1f3dc3016c3d19f43972eb.

A working version of the code is also available utilizing the SEALSQ Secure element https://www.sealsq.com/semiconductors/vaultic-secure-elements/vaultic-40x.

3.4.7 NXP Semiconductors

NXP Semiconductors focuses on secure connectivity solutions for embedded applications, NXP is impacting the automotive, industrial, and IoT, mobile, and communication infrastructure markets. Built on more than 60 years of combined experience and expertise, the company has approximately 31,000 employees in more than 30 countries. Find out more at https://www.nxp.com/.

3.4.7.1 EdgeLock SE050 secure element

The EdgeLock SE050 secure element (SE) product family offers strong protection against the latest attack scenarios and an extended feature set for a broad range of IoT use cases. This ready-to-use secure element for IoT devices provides a root of trust at the silicon level and delivers real end-to-end security – from edge to cloud – with a comprehensive software package for integration into any type of device.

3.4.7.2 EdgeLock 2GO

EdgeLock 2GO is the NXP service platform designed for easy and secure deployment and management of IoT devices. This flexible IoT service platform lets the device manufacturers and service providers choose the appropriate options to optimize costs while benefiting from an advanced level of device security. The EdgeLock 2GO service provisions the cryptographic keys and certificates into the hardware root of trust of the IoT devices and simplifies the onboarding of the devices to the cloud.

3.4.7.3 i.MX 8M family

The i.MX 8M family of applications processors based on Arm® Cortex®-A53 and Cortex-M4 cores provide advanced audio, voice, and video processing for applications that scale from consumer home audio to industrial building automation and mobile computers. It includes support for secure boot, secure debug, and lifecycle management, as well as integrated cryptographic accelerators. The development boards and Linux Board Support Package enablement provide out-of-the-box integration with an external SE050 secure element.

3.4.8 Open Connectivity Foundation (OCF)

OCF is a standards-developing organization that has had contributions and participation from over 450+ member organizations representing the full spectrum of the IoT ecosystem, from chip makers to consumer electronics manufacturers, silicon enablement software platform and service providers, and network operators. The OCF specification is an International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) internationally recognized standard that was built in tandem with an open-source reference implementation called IoTivity. Additionally, OCF provides an in-depth testing and certification program.

3.4.8.1 IoTivity

OCF has contributed open-source code from IoTivity that demonstrates the advantage of secure network-layer onboarding and implements the WFA's Easy Connect to power a seamless bootstrapping of secure and trusted application-layer onboarding of IoT devices with minimal user interaction.

This code includes the interaction layer, called the OCF Diplomat, which handles secure communication between the DPP-enabled access point and the OCF application layer. The OCF onboarding tool (OBT) is used to configure and provision devices with operational credentials. The OCF reference implementation of a basic lamp is used to demonstrate both network- and application-layer onboarding and to show that once onboarded and provisioned, the OBT can securely interact with the lamp.

3.4.9 Sandelman Software Works

Sandelman Software Works (SSW) provides consulting and software design services in the areas of systems and network security. A complete stack company, SSW provides consulting and design services from the hardware driver level up to Internet Protocol Security (IPsec), Transport Layer Security (TLS), and cloud database optimization. SSW has been involved with the IETF since the 1990s, now dealing with the difficult problem of providing security for IoT systems. SSW leads standardization efforts through a combination of running code and rough consensus.

3.4.9.1 Minerva Highway IoT Network-Layer Onboarding and Lifecycle Management System

The Highway component is a cloud-native component operated by the device manufacturer (or its authorized designate). It provides the Request for Comments (RFC) 8995 [7] specified Manufacturer Authorized Signing Authority (MASA) for the BRSKI onboarding mechanism.

Highway is an asset manager for IoT devices. In its asset database it maintains an inventory of devices that have been manufactured, what type they are, and who the current owner of the device is (if it has been sold). Highway does this by taking control of the complete identity lifecycle of the device. It can aid in provisioning new device identity certificates (IDevIDs) by collecting Certificate Signing Requests and returning certificates, or by generating the new identities itself. This is consistent with Section 4.1.2.1 (On-device private key generation) and Section 4.1.2.2 (Off-device private key generation) of https://www.ietf.org/archive/id/draft-irtf-t2trg-taxonomy-manufacturer-anchors-00.html.

Highway can act as a standalone three-level private-public key infrastructure (PKI). Integrations with Automatic Certificate Management Environment (RFC 8555) allow it to provision certificates from an external PKI using the DNS-01 challenge in Section 8.4 of https://www.rfc-editor.org/rfc/rfc8555.html#section-8.4. Hardware integrations allow for the private key operations to be moved out of the main CPU. However, the needs of a busy production line in a factory would require continuous access to the hardware offload.

In practice, customers put the subordinate CA into Highway, which it needs to sign new IDevIDs, and put the trust anchor private CA into a hardware security module (HSM).

Highway provides a BRSKI-MASA interface running on a public TCP/HTTPS port (usually 443 or 9443). This service requires access to the private key associated to the anchor that has been “baked into” the Pledge device during manufacturing. The Highway instance that speaks to the world in this way does not have to be the same instance that signs IDevID certificates, and there are significant security advantages to separating them. Both instances do need access to the same database servers, and there are a variety of database replication techniques that can be used to improve resilience and security.

As IDevIDs do not expire, Highway does not presently include any mechanism to revoke IDevIDs, nor does it provide Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP). It is unclear how those mechanisms can work in practice.

Highway supports two models. In the Sales Integration model, the intended owner is known in advance. This model requires customer-specific integrations, which often occur at the database level through views or other SQL tools. In the trust on first use (TOFU) model, the first customer to claim a product becomes its owner.

3.4.10 SEALSQ, a subsidiary of WISeKey

WISeKey International Holding Ltd. (WISeKey) is a cybersecurity company that deploys digital identity ecosystems and secures IoT solution platforms. It operates as a Swiss-based holding company through several operational subsidiaries, each dedicated to specific aspects of its technology portfolio.

SEALSQ is the subsidiary of the group that focuses on designing and selling secure microcontrollers, PKI, and identity provisioning services while developing post-quantum technology hardware and software products. SEALSQ products and solutions are used across a variety of applications today, from multi-factor authentication devices, home automation systems, and network infrastructure, to automotive, industrial automation, and control systems.

3.4.11 VaultIC408

The VaultIC408 secure element combines hardware-based key storage with cryptographic accelerators to provide a wide array of cryptographic features including identity, authentication, encryption, key agreement, and data integrity. It protects against hardware attacks such as micro-probing and side channels.

The fundamental cryptography of the VaultIC family includes NIST-recommended algorithms and key lengths. Each of these algorithms, Elliptic Curve Cryptography (ECC), Rivest-Shamir-Adleman (RSA), and Advanced Encryption Standard (AES), is implemented on-chip and uses on-chip storage of the secret key material so the secrets are always protected in the secure hardware.

The secure storage and cryptographic acceleration support use cases like network and IoT end node security, platform security, secure boot, secure firmware download, secure communication or TLS, data confidentiality, encryption key storage, and data integrity.

3.4.11.1 INeS Certificate Management System (CMS)

SEALSQ's portfolio includes INeS, a managed PKI-as-a-service solution. INeS leverages the WISeKey Webtrust-accredited trust services platform, a Matter approved Product Attestation Authority (PAA), and custom CAs. These PKI technologies support large-scale IoT deployments, where IoT endpoints will require certificates to establish their identities. The INeS CMS platform provides a secure, scalable, and manageable trust model.

INeS CMS provides certificate management, CA management, public cloud integration and automation, role-based access control (RBAC), and APIs for custom implementations.

3.4.12 Silicon Labs

Silicon Labs provides products in the area of secure, intelligent wireless technology for a more connected world. Securing IoT is challenging. It's also mission critical. The challenge of protecting connected devices against frequently surfacing IoT security vulnerabilities follows device makers throughout the entire product lifecycle. Protecting products in a connected world is a necessity as customer data and modern online business models are increasingly targets for costly hacks and corporate brand damage. To stay secure, device makers need an underlying security platform in the hardware, software, network, and cloud. Silicon Labs offers security products with features that address escalating IoT threats, with the goal of reducing the risk of IoT ecosystem security breaches and the compromise of intellectual property and revenue loss from counterfeiting.

For this project, Silicon Labs has provided a host platform for the OpenThread border router (OTBR), a Thread radio transceiver, and an IoT device to be onboarded to the AWS cloud service and that communicates using the Thread wireless protocol.

3.4.12.1 OpenThread Border Router Platform

A Raspberry Pi serves as host platform for the OTBR. The OTBR forms a Thread network and acts as a bridge between the Thread network and the public internet, allowing the IoT device that communicates using the Thread wireless protocol and that is to be onboarded communicate with cloud services. The OTBR's connection to the internet can be made through either Wi-Fi or ethernet. Connection to the SLWSTK6023A (see Section 3.4.12.2) is made through a USB serial port.

3.4.12.2 SLWSTK6023A Thread Radio Transceiver

The SLWSTK6023A (Wireless starter kit) acts as a Thread radio transceiver or radio coprocessor (RCP). This allows the OTBR host platform to form and communicate with a Thread network.

3.4.12.3 xG24-DK2601B Thread "End" Device

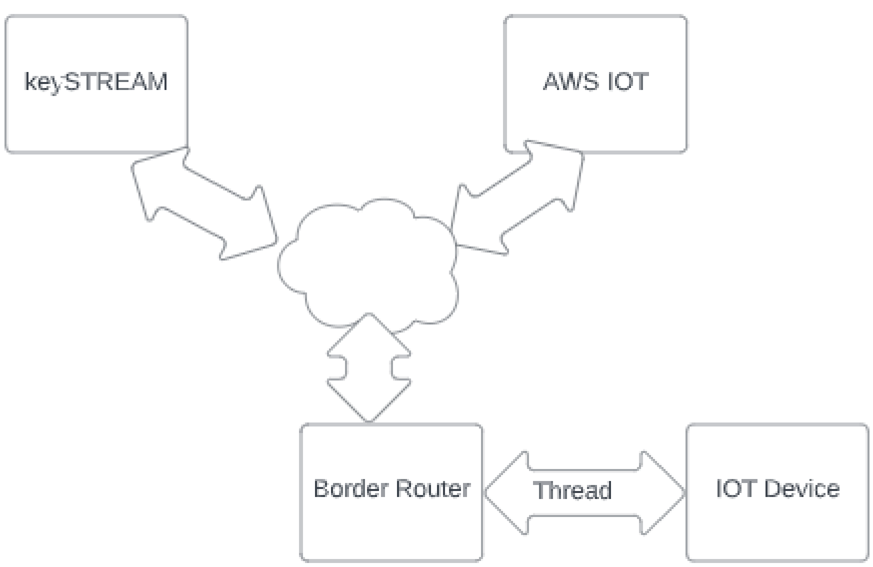
The xG24-DK2601B is the IoT device that is to be onboarded to the cloud service (AWS). It communicates using the Thread wireless protocol. Communication is bridged between the Thread network and the internet by the OTBR.

3.4.12.4 Kudelski IoT keySTREAM™

The Kudelski IoT keySTREAM solution is described more fully in Section 3.4.5.1. It is a cloud service capable of verifying the hardware-based secure identity certificate chain associated with the xG24-DK2601B component described in Section 3.4.12.3 and delivering a new certificate chain that can be refreshed or revoked as needed to assist with lifecycle management. The certificate chain is used to authenticate the xG24-DK2601B device to the cloud service (AWS).

Figure 3-2 shows the relationships among the components provided by Silicon Labs and Kudelski that support the trusted application-layer onboarding of an IoT device that communicates via the Thread protocol to AWS IoT.

Figure 3-2 Components for Onboarding an IoT Device that Communicates Using Thread to AWS IoT

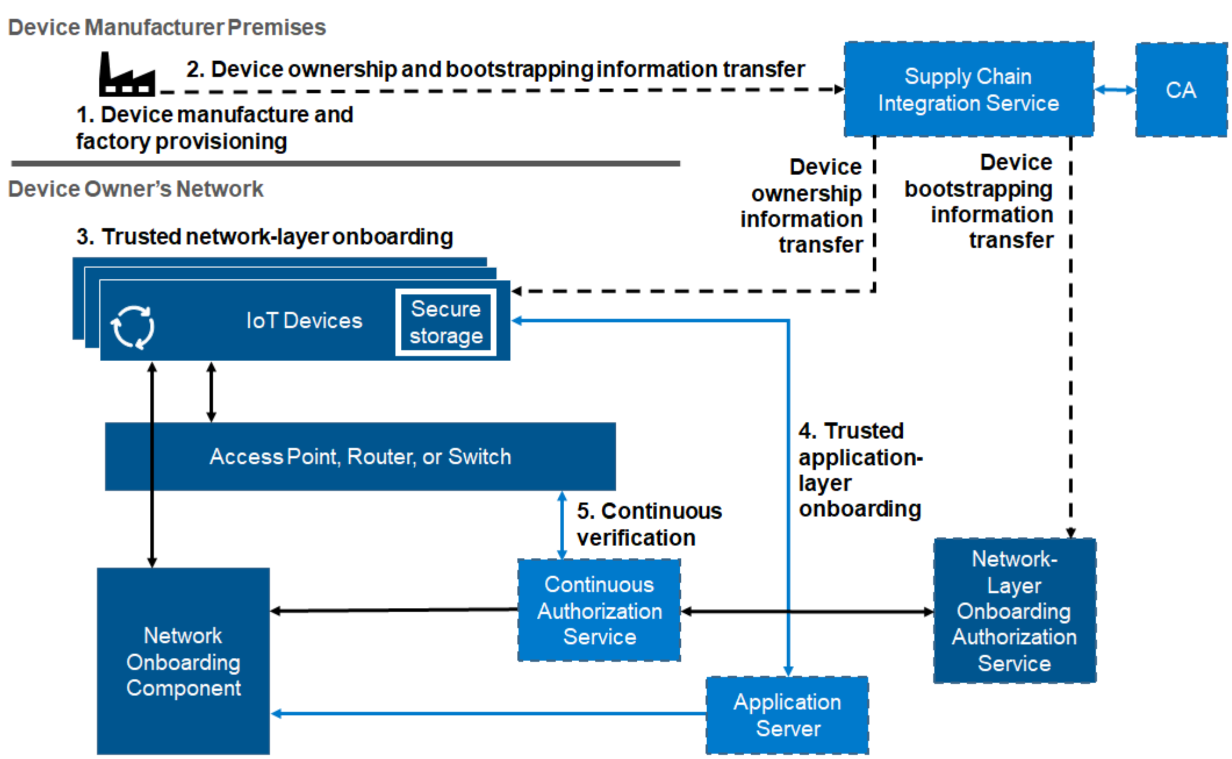


4 Reference Architecture

Figure 4-1 depicts the reference architecture to demonstrate trusted IoT device network-layer onboarding and lifecycle management used throughout this Practice Guide. This architecture shows a high-level, protocol-agnostic, and generic approach to trusted network-layer onboarding. It represents the basic components and processes, regardless of the network-layer onboarding protocol used and the particular device lifecycle management activities supported.

When implementing this architecture, an organization can follow different steps and use different components. The exact steps that are performed may not be in the same order as the steps in the logical reference architecture, and they may use components that do not have a one-to-one correspondence with the logical components in the logical reference architecture. In Appendices C, D, E, F and G we present the architectures for builds 1, 2, 3, 4 and 5, each of which is an instantiation of this logical reference architecture. Those build-specific architectures are more detailed and are described in terms of specific collaborator components and trusted network-layer onboarding protocols.

Figure 4-1 Trusted IoT Device Network-Layer Onboarding and Lifecycle Management Logical Reference Architecture



There are five high-level processes to carry out this architecture, as labeled in Figure 4-1. These five processes are as follows:

1. **Device manufacture and factory provisioning** – the activities that the IoT device manufacturer performs to prepare the IoT device so that it is capable of network- and application-layer onboarding (Figure 4-2, Section 4.1).

2. **Device ownership and bootstrapping information transfer** – the transfer of IoT device ownership and bootstrapping information from the manufacturer to the device and/or the device's owner that enables the owner or an entity authorized by the owner to onboard the device securely. The component in Figure 4-1 labeled "Supply Chain Integration Service" represents the mechanism used to accomplish this information transfer (Figure 4-3, Section 4.2).

3. **Trusted network-layer onboarding** – the interactions that occur between the network-layer onboarding component and the IoT device to mutually authenticate, confirm authorization, establish a secure channel, and provision the device with its network credentials (Figure 4-4, Section 4.3).

4. **Trusted application-layer onboarding** – the interactions that occur between a trusted application server and the IoT device to mutually authenticate, establish a secure channel, and provision the device with application-layer credentials (Figure 4-5, Section 4.4).

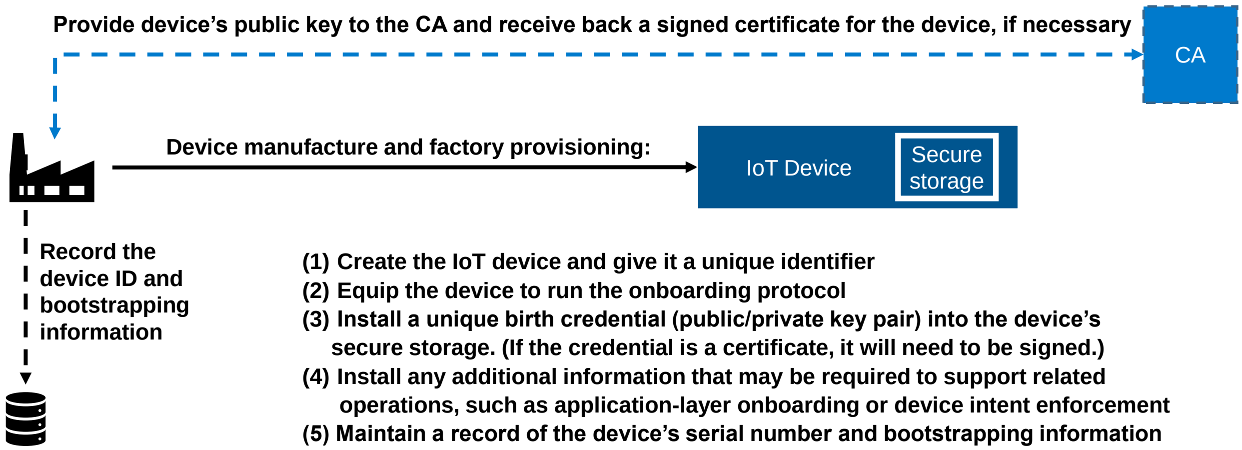
5. **Continuous verification** – ongoing, policy-based verification and authorization checks on the IoT device to support device lifecycle monitoring and control (Figure 4-6, Section 4.5).

Figure 4-1 uses two colors. The dark-blue components are central to supporting trusted network-layer onboarding itself. The light-blue components support the other aspects of the architecture. Each of the five processes is explained in more detail in the subsections below.

4.1 Device Manufacture and Factory Provisioning Process

Figure 4-2 depicts the device manufacture and factory provisioning process in more detail. As shown in Figure 4-2, the manufacturer is responsible for creating the IoT device and provisioning it with the necessary hardware, software, and birth credentials so that it is capable of network-layer onboarding. The IoT device should be manufactured with a secure root of trust as a best practice, possibly as part of a secure manufacturing process, particularly when outsourced. Visibility and control over the provisioning process and manufacturing supply chain, particularly for outsourced manufacturing, is critical in order to mitigate the risk of compromise in the supply chain, which could lead to the introduction of compromised devices. The CA component is shown in light blue in Figure 4-2 because its use is optional and depends on the type of credential that is being provisioned to the device (i.e., whether it is an 802.1AR certificate).

Figure 4-2 IoT Device Manufacture and Factory Provisioning Process



At a high level, the steps that the manufacturer or an integrator performs as part of this preparation process, as shown in Figure 4-2, are as follows:

1. Create the IoT device and assign it a unique identifier (e.g., a serial number). Equip the device with secure storage.

2. Equip the device to run a specific network-layer onboarding protocol (e.g., Wi-Fi Easy Connect, BRSKI, Thread Mesh Commissioning Protocol (MeshCoP) [8]). This step includes ensuring that the device has the software/firmware needed to run the onboarding protocol as well as any additional information that may be required.

3. Generate or install the device's unique birth credential into the device's secure storage. [Note: using a secure element that has the ability to autonomously generate private/public root key pairs is inherently more secure than performing credential injection, which has the potential to expose the private key.] The birth credential includes information that must be kept secret (i.e., the device's private key) because it is what enables the device's identity to be authenticated. The contents of the birth credential will depend on what network-layer onboarding protocol the device supports. For example:

    a. If the device runs the Wi-Fi Easy Connect protocol, its birth credential will take the form of a unique private key, which has an associated DPP URI that includes the corresponding public key and possibly additional information such as Wi-Fi channel and serial number.

    b. If the device runs the BRSKI protocol, its birth credential takes the form of an 802.1AR certificate that gets installed as the device's IDevID and corresponding private key. The IDevID includes the device's public key, the location of the MASA, and trust anchors that can be used to verify vouchers signed by the MASA. The 802.1AR certificate needs to be signed by a trusted signing authority prior to installation, as shown in Figure 4-2.

4. Install any additional information that may be required to support related capabilities that are enabled by network-layer onboarding. The specific contents of the information that gets

installed on the device will vary according to what capabilities it is intended to support. For example, if the device supports:

a. **streamlined application-layer onboarding** (see Section 3.3.2), then the bootstrapping information that is required to enable the device and a trusted application server to find and mutually authenticate each other and establish a secure association will be stored on the device. This is so it can be sent to the network during network-layer onboarding and used to automatically perform application-layer onboarding after the device has securely connected to the network. The Wi-Fi Easy Connect protocol, for example, can include such application-layer bootstrapping information as third-party information in its protocol exchange with the network, and Build 2 (i.e., the Wi-Fi Easy Connect, CableLabs, OCF build) demonstrates use of this mechanism to support streamlined application-layer onboarding.

   Note, however, that a device may still be capable of performing independent [see Section 3.3.2] application-layer onboarding even if the application-layer onboarding information is not exchanged as part of the network-layer onboarding protocol. The application that is installed on the device, i.e., the application that the device executes to fulfill its purpose, may include application-layer bootstrapping information that enables it to perform application-layer onboarding when it begins executing. Build 1 (i.e., the Wi-Fi Easy Connect, Aruba/HPE build) demonstrates independent application-layer onboarding.

b. **device communications intent**, then the URI required to enable the network to locate the device's intent information may be stored on the device so that it can be sent to the network during network-layer onboarding. After the device has securely connected to the network, the network can use this device communications intent information to ensure that the device sends and receives traffic only from authorized locations.

5. Maintain a record of the device's serial number (or other uniquely identifying information) and the device's bootstrapping information. The manufacturer will take note of the device's ID and its bootstrapping information and store these. Eventually, when the device is sold, the manufacturer will need to provide the device's owner with its bootstrapping information. The contents of the device's bootstrapping information will depend on what network-layer onboarding protocol the device supports. For example:

   a. If the device runs the Wi-Fi Easy Connect protocol, its bootstrapping information is the DPP URI that is associated with its private key.

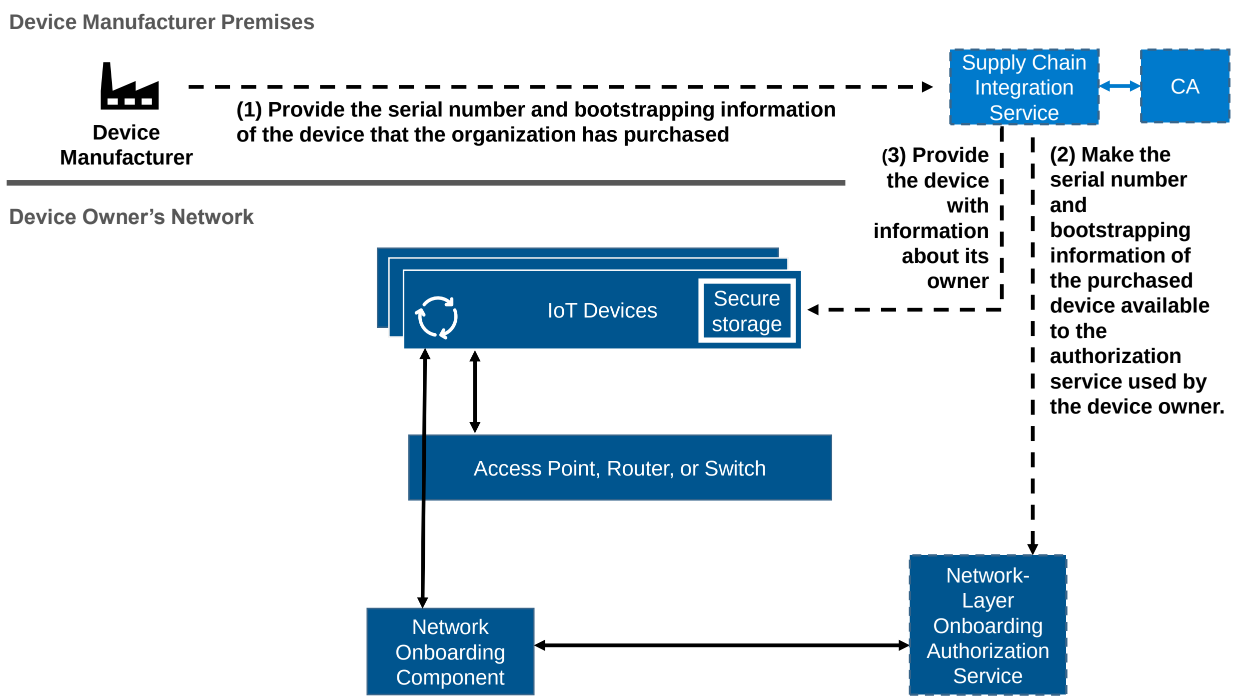
   b. If the device runs the BRSKI protocol, its bootstrapping information is its 802.1AR certificate.

4.2 Device Ownership and Bootstrapping Information Transfer Process

Figure 4-3 depicts the activities that are performed to transfer device bootstrapping information from the device manufacturer to the device owner, as well as to transfer device ownership information to the device itself, if appropriate. A high-level summary of these activities is described in the steps labeled A, B, and C.

The figure uses two colors. The dark-blue components are those used in the network-layer onboarding process. They are the same components as those depicted in the trusted network-layer onboarding process diagram provided in Figure 4-4. The light-blue components and their accompanying steps depict the portion of the diagram that is specific to device ownership and bootstrapping information transfer activities.

Figure 4-3 Device Ownership and Bootstrapping Information Transfer Process



These steps are as follows:

1. The device manufacturer makes the device serial number, bootstrapping information, and ownership information available so that the organization or individual who has purchased the device will have the device's serial number and bootstrapping information, and the device itself can be informed of who its owner is. In Figure 4-3, the manufacturer is shown sending this information to the supply chain integration service, which ensures that the necessary information ultimately reaches the device owner's authorization service as well as the device itself, if appropriate. (This description of the process is deliberately simple in order to enable it to be general enough that it applies to a variety of network-layer onboarding protocols.) In reality, the supply chain integration service mechanism for forwarding this bootstrapping information from the manufacturer to the owner may take many forms. For example, when BRSKI is used, the manufacturer sends the device serial number and bootstrapping information to a MASA that both the device and its owner trust. When other network-layer onboarding protocols are used, the device manufacturer may provide the device owner with this bootstrapping information directly by uploading this information to the owner's portion of a

trusted cloud. Such a mechanism is useful for the case in which the owner is a large enterprise that has made a bulk purchase of many IoT devices. In this case, the manufacturer can upload the information for hundreds or thousands of IoT devices to the supply chain integration service at once. We call this the enterprise use case. Alternatively, the device manufacturer may provide this information to the device owner indirectly by including it on or in the packaging of an IoT device that is sold at retail. We call this the consumer use case.

The contents of the device bootstrapping information will also vary according to the network-layer onboarding protocol that the device supports. For example, if the device supports the Wi-Fi Easy Connect network-layer onboarding protocol, the bootstrapping information will consist of the device's DPP URI. If the device supports the BRSKI network-layer onboarding protocol, bootstrapping information will consist of the device's IDevID (i.e., its 802.1AR certificate).

2. The supply chain integration service forwards the device serial number and bootstrapping information to an authorization service that has connectivity to the network-layer onboarding component that will onboard the device (i.e., to a network-layer onboarding component that belongs either to the device owner or to an entity that the device owner has authorized to onboard the device). The network-layer onboarding component will use the device's bootstrapping information to authenticate the device and verify that it is expected and authorized to be onboarded to the network. Again, this forwarding may take many forms, e.g., enterprise use case or consumer use case, and use a variety of different mechanisms within each use case type, e.g., information moved from one location to another in the device owner's portion of a trusted cloud, information transferred via a standardized protocol operating between the MASA and the onboarding network's domain registrar, or information scanned from a QR code on device packaging using a mobile app. In the case in which BRSKI is used, a certificate authority is consulted to help validate the signature of the 802.1AR certificate that comprises the device bootstrapping information.

3. The supply chain integration service may also provide the device with information about who its owner is. Knowing who its owner is enables the device to ensure that the network that is trying to onboard it is authorized to do so, because it is assumed that if a network owns a device, it is authorized to onboard it. The mechanisms for providing the device with assurance that the network that is trying to onboard it is authorized to do so can take a variety of forms, depending on the network-layer onboarding protocol being used. For example, if the Wi-Fi Easy Connect protocol is being used, then if an entity is in possession of the device's public key, that entity is assumed to be authorized to onboard the device. If BRSKI is being used, the device will be provided with a signed voucher verifying that the network that is trying to onboard the device is authorized to do so. The voucher is signed by the MASA. Because the device manufacturer has installed trust anchors for the MASA onto the device, the device trusts the MASA. It is also able to verify the MASA's signature.

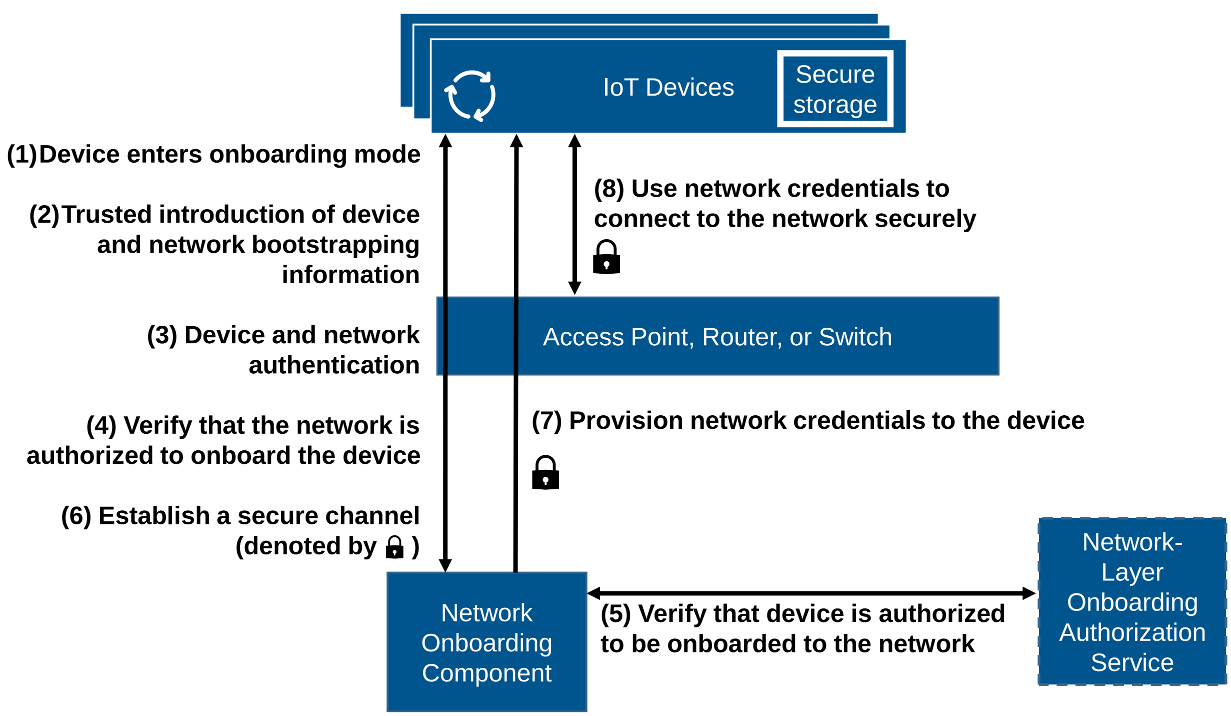
(Note: In this document, for the sake of simplicity, we often refer to the network that is authorized to onboard a device as the device owner's network. In reality, it may not always be the case that the device's owner also owns the network to which the device is being onboarded. While it is assumed that a network that owns a device is authorized to onboard it, and the device and the onboarding network are often owned by the same entity, common ownership is not a requirement. The network that is onboarding a device does not have to be the owner of that device. The network owner may permit devices that it does not own to be onboarded to the network. In order for such a device to be onboarded, the network owner must be in possession of the device's bootstrapping information. By accepting the bootstrapping information, the network owner is implicitly authorizing the device to be onboarded to its network. Conversely, a device may permit itself to be onboarded to a network that is not owned by the device's owner. A device owner that wants to authorize a network to onboard the device needs to ensure that the device trusts the onboarding network. The specific mechanism for accomplishing this will vary according to the network-layer onboarding protocol being used. When the Wi-Fi Easy Connect protocol is being used, simply providing the network with the device's public key is sufficient to authorize the network to onboard the device. When BRSKI is being used, the voucher that the MASA provides to the device must authorize the network to onboard it.)

Authentication of the network by the device may also take a variety of forms. These may range from simply trusting the person who is onboarding the device to onboard it to the correct network, to providing the IoT device with the network's public key.

4.3 Trusted Network-Layer Onboarding Process

Figure 4-4 depicts the trusted network-layer onboarding process in more detail. It shows the interactions that occur between the network-layer onboarding component and the IoT device to mutually authenticate, confirm that the device is authorized to be onboarded to the network, confirm that the network is authorized to onboard the device, establish a secure channel, and provision the device with its network credentials.

Figure 4-4 Trusted Network-Layer Onboarding Process



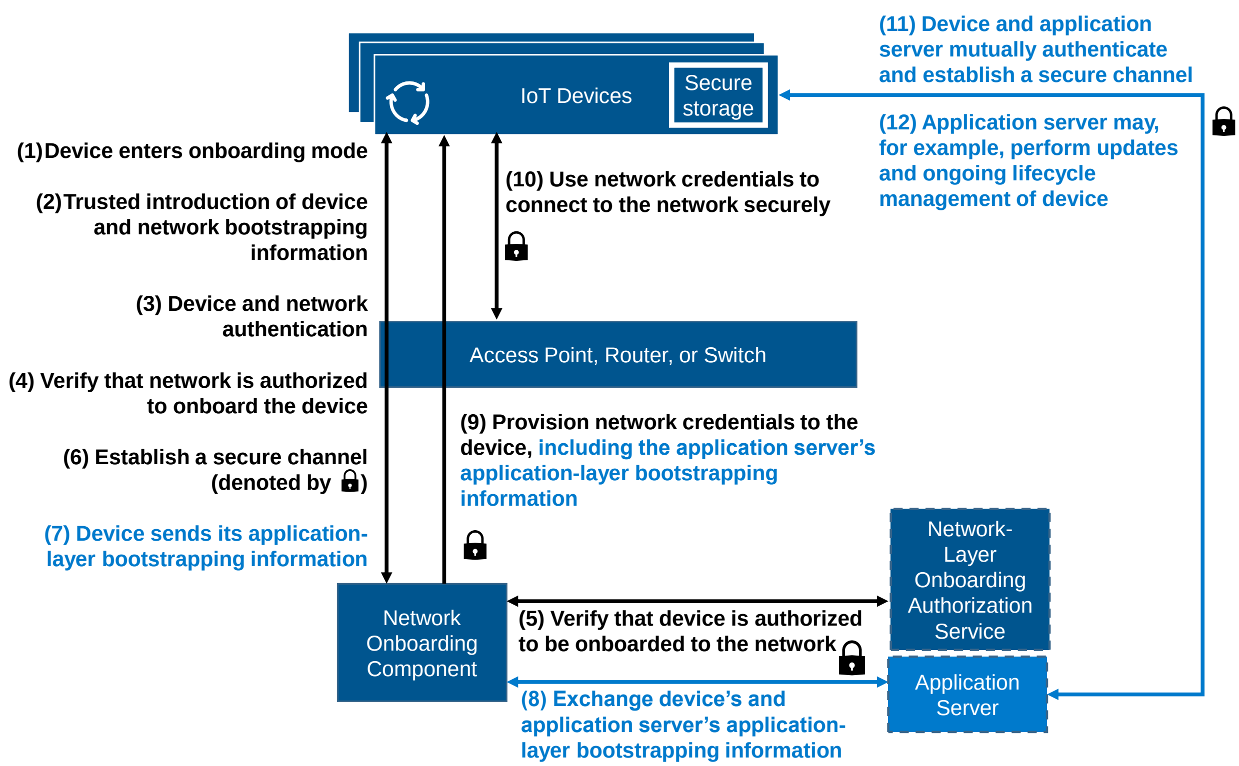
The numbered arrows in the diagram are intended to provide a high-level summary of the network-layer onboarding steps. These steps are assumed to occur after any device bootstrapping information and ownership transfer activities (as described in the previous section) that may need to be performed. The steps of the trusted network-layer onboarding process are as follows:

1. The IoT device to be onboarded is placed in onboarding mode, i.e., it is put into a state such that it is actively listening for and/or sending initial onboarding protocol messages.
2. Any required device bootstrapping information that has not already been provided to the network and any required network bootstrapping information that has not already been provided to the device are introduced in a trusted manner.
3. Using the device and network bootstrapping information that has been provided, the network authenticates the identity of the IoT device (e.g., by ensuring that the IoT device is in possession of the private key that corresponds with the public key for the device that was provided as part of the device's bootstrapping information), and the IoT device authenticates the identity of the network (e.g., by ensuring that the network is in possession of the private key that corresponds with the public key for the network that was provided as part of the network's bootstrapping information).
4. The device verifies that the network is authorized to onboard it. For example, the device may verify that it and the network are owned by the same entity, and therefore, assume that the network is authorized to onboard it.
5. The network onboarding component consults the network-layer onboarding authorization service to verify that the device is authorized to be onboarded to the network. For example, the network-layer authorization service can confirm that the device is owned by the network and is on the list of devices authorized to be onboarded.
6. A secure (i.e., encrypted) channel is established between the network onboarding component and the device.
7. The network onboarding component uses the secure channel that it has established with the device to confidentially send the device its unique network credentials.
8. The device uses its newly provisioned network credentials to establish secure connectivity to the network. The access point, router, or switch validates the device's credentials in this step. The mechanism it uses to do so varies depending on the implementation and is not depicted in Figure 4-4.

4.4 Trusted Application-Layer Onboarding Process

Figure 4-5 depicts the trusted application-layer onboarding process as enabled by the streamlined application-layer onboarding mechanism. As defined in Section 3.3.2, streamlined application-layer onboarding occurs after network-layer onboarding and depends upon and is enabled by it. The figure uses two colors. The dark-blue components are those used in the network-layer onboarding process. They and their accompanying steps (written in black font) are identical to those found in the trusted network-layer onboarding process diagram provided in Figure 4-4. The light-blue component and its accompanying steps (written in light-blue font) depict the portion of the diagram that is specific to streamlined application-layer onboarding.

Figure 4-5 Trusted Streamlined Application-Layer Onboarding Process



As is the case with Figure 4-4, the steps in this diagram are assumed to occur after any device ownership and bootstrapping information transfer activities that may need to be performed. Steps 1-6 in this figure are identical to Steps 1-6 in the trusted network-layer onboarding diagram of Figure 4-4, but steps 7 and 8 are different. With the completion of steps 1-6 in Figure 4-5, a secure channel has been established between the IoT device and the network-layer onboarding component. However, the device does not get provisioned with its network-layer credentials until step 9. To support streamlined application-layer onboarding, additional steps are required. Steps 1-12 are as follows:

1. The IoT device to be onboarded is placed in onboarding mode, i.e., it is put into a state such that it is actively listening for and/or sending initial onboarding protocol messages.
2. Any required device bootstrapping information that has not already been provided to the network and any required network bootstrapping information that has not already been provided to the device are introduced in a trusted manner.
3. Using the device and network bootstrapping information that has been provided, the network authenticates the identity of the IoT device (e.g., by ensuring that the IoT device is in possession of the private key that corresponds with the public key for the device that was provided as part of the device's bootstrapping information), and the IoT device authenticates the identity of the network (e.g., by ensuring that the network is in possession of the private key that corresponds

with the public key for the network that was provided as part of the network's bootstrapping information).

4. The device verifies that the network is authorized to onboard it. For example, the device may verify that it and the network are owned by the same entity, and therefore, assume that the network is authorized to onboard it.

5. The network onboarding component consults the network-layer onboarding authorization service to verify that the device is authorized to be onboarded to the network. For example, the network-layer authorization service can confirm that the device is owned by the network and is on the list of devices authorized to be onboarded.

6. A secure (i.e., encrypted) channel is established between the network onboarding component and the device.

7. The device sends its application-layer bootstrapping information to the network onboarding component. Just as the network required the trusted introduction of device network-layer bootstrapping information in order to enable the network to authenticate the device and ensure that the device was authorized to be network-layer onboarded, the application server requires the trusted introduction of device application-layer bootstrapping information to enable the application server to authenticate the device at the application layer and ensure that the device is authorized to be application-layer onboarded. Because this application-layer bootstrapping information is being sent over a secure channel, its integrity and confidentiality are ensured.

8. The network onboarding component forwards the device's application-layer bootstrapping information to the application server. In response, the application server provides its application-layer bootstrapping information to the network-layer onboarding component for eventual forwarding to the IoT device. The IoT device needs the application server's bootstrapping information to enable the device to authenticate the application server and ensure that it is authorized to application-layer onboard the device.

9. The network onboarding component uses the secure channel that it has established with the IoT device to confidentially send the device its unique network credentials. Along with these network credentials, the network onboarding component also sends the IoT device the application server's bootstrapping information. Because the application server's bootstrapping information is being sent over a secure channel, its integrity and confidentiality are ensured.z

10. The device uses its newly provisioned network credentials to establish secure connectivity to the network.

11. Using the device and application server application-layer bootstrapping information that has already been exchanged in a trusted manner, the application server authenticates the identity of the IoT device and the IoT device authenticates the identity of the application server. Then they establish a secure (i.e., encrypted) channel.

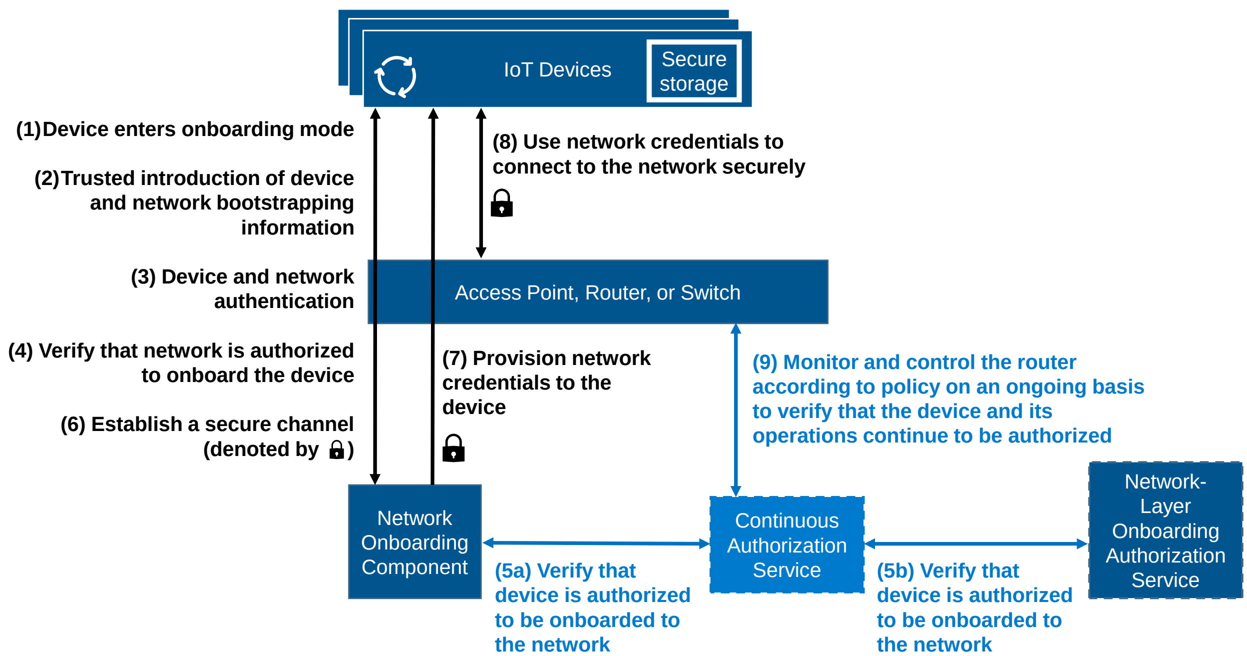
12. The application server application layer onboards the IoT device. This application-layer onboarding process may take a variety of forms. For example, the application server may download an application to the device for the device to execute. It may associate the device

with a trusted lifecycle management service that performs ongoing updates of the IoT device to patch it as needed to ensure that the device remains compliant with policy.

4.5 Continuous Verification

Figure 4-6 depicts the steps that are performed to support continuous verification. The figure uses two colors. The light-blue component and its accompanying steps (written in light-blue font) depict the portion of the diagram that is specific to continuous authorization. The dark-blue components are those used in the network-layer onboarding process. They and their accompanying steps (written in black font) are identical to those found in the trusted network-layer onboarding process diagram provided in Figure 4-4, except for step 5, *Verify that device is authorized to be onboarded to the network*.

Figure 4-6 Continuous Verification



When continuous verification is being supported, step 5 is broken into two separate steps, as shown in Figure 4-6. Instead of the network onboarding component directly contacting the network-layer onboarding authorization service to see if the device is owned by the network and on the list of devices authorized to be onboarded (as shown in the trusted network-layer onboarding architecture depicted in Figure 4-4), a set of other enterprise policies may also be applied to determine if the device is authorized to be onboarded. The application of these policies is represented by the insertion of the Continuous Authorization Service (CAS) component in the middle of the exchange between the network onboarding component and the network-layer onboarding authorization service.

For example, the CAS may have received external threat information indicating that certain device types have a vulnerability. If so, when the CAS receives a request from the network-layer onboarding component to verify that a device of this type is authorized to be onboarded to the network (Step 5a), it would immediately respond to the network-layer onboarding component that the device is not authorized to be onboarded to the network. If the CAS has not received any such threat information about the device and it checks all its policies and determines that the device should be permitted to be onboarded, it will forward the request to the network-layer onboarding authorization service (Step 5b) and receive a response (Step 5b) that it will forward to the network onboarding component (Step 5a).

As depicted by Step 9, the CAS also continues to operate after the device connects to the network and executes its application. The CAS performs asynchronous calls to the network router to monitor the device on an ongoing basis, providing policy-based verification and authorization checks on the device throughout its lifecycle.

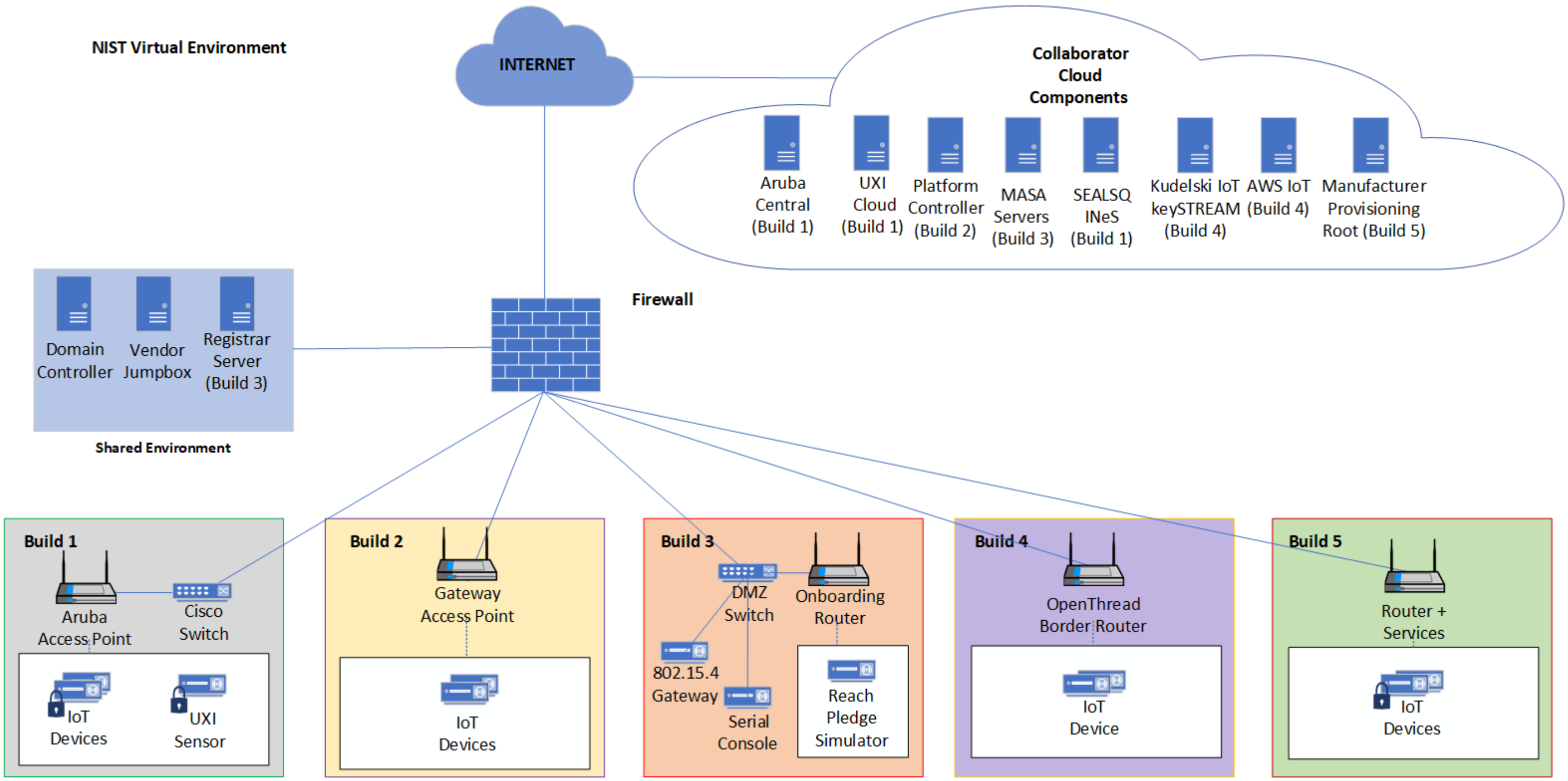
5 Laboratory Physical Architecture

Figure 5-1 depicts the high-level physical architecture of the NCCoE IoT Onboarding laboratory environment in which the five trusted IoT device network-layer onboarding project builds, and the factory provisioning builds are being implemented. The NCCoE provides virtual machine (VM) resources and physical infrastructure for the IoT Onboarding lab. As depicted, the NCCoE IoT Onboarding laboratory hosts collaborator hardware and software for the builds. The NCCoE also provides connectivity from the IoT Onboarding lab to the NIST Data Center, which provides connectivity to the internet and public IP spaces (both IPv4 and IPv6). Access to and from the NCCoE network is protected by a firewall.

Access to and from the IoT Onboarding lab is protected by a pfSense firewall, represented by the brick box icon in Figure 5-1. This firewall has both IPv4 and IPv6 (dual stack) configured. The IoT Onboarding lab network infrastructure includes a shared virtual environment that houses a domain controller and a vendor jumpbox. These components are used across builds where applicable. It also contains five independent virtual LANs, each of which houses a different trusted network-layer onboarding build.

The IoT Onboarding laboratory network has access to cloud components and services provided by the collaborators, all of which are available via the internet. These components and services include Aruba Central and the UXI Cloud (Build 1), SEALSQ INeS (Build 1), Platform Controller (Build 2), a MASA server (Build 3), Kudelski IoT keySTREAM application-layer onboarding service and AWS IoT (Build 4), and a Manufacturer Provisioning Root (Build 5).

1435 **Figure 5-1 NCCoE IoT Onboarding Laboratory Physical Architecture**



All five network-layer onboarding laboratory environments, as depicted in the diagram, have been installed:

- The Build 1 (i.e., the Wi-Fi Easy Connect, Aruba/HPE build) network infrastructure within the NCCoE lab consists of two components: the Aruba Access Point and the Cisco Switch. Build 1 also requires support from Aruba Central for network-layer onboarding and the UXI Cloud for application-layer onboarding. These components are in the cloud and accessed via the internet. The IoT devices that are onboarded using Build 1 include the UXI Sensor and the Raspberry Pi.
- The Build 2 (i.e., the Wi-Fi Easy Connect, CableLabs, OCF build) network infrastructure within the NCCoE lab consists of a single component: the Gateway Access Point. Build 2 requires support from the Platform Controller, which also hosts the IoTivity Cloud Service. The IoT devices that are onboarded using Build 2 include three Raspberry Pis.
- The Build 3 (i.e., the BRSKI, Sandelman Software Works build) network infrastructure components within the NCCoE lab include a Wi-Fi capable home router (including Join Proxy), a DMZ switch (for management), and an ESP32A Xtensa board acting as a Wi-Fi IoT device, as well as an nRF52840 board acting as an IEEE 802.15.4 device. A management system on a BeagleBone Green serves as a serial console. A registrar server has been deployed as a virtual appliance on the NCCoE private cloud system. Build 3 also requires support from a MASA server which is accessed via the internet. In addition, a Raspberry Pi 3 provides an ethernet/802.15.4 gateway, as well as a test platform.
- The Build 4 (i.e., the Thread, Silicon Labs, Kudelski IoT build) network infrastructure components within the NCCoE lab include an Open Thread Border Router, which is implemented using a Raspberry Pi, and a Silicon Labs Gecko Wireless Starter Kit, which acts as an 802.15.4 antenna. Build 4 also requires support from the Kudelski IoT keySTREAM service, which is in the cloud and accessed via the internet. The IoT device that is onboarded in Build 4 is the Silicon Labs Dev Kit (BRD2601A) with an EFR32MG24 System-on-Chip (SoC). The application service to which it onboards is AWS IoT.
- The Build 5 (i.e., the BRSKI over Wi-Fi, NquiringMinds build) includes 2 Raspberry Pi 4Bs running a Linux operating system. One Raspberry Pi acts as the pledge (or IoT Device) with an Infineon TPM connected. The other acts as the router, registrar and MASA all in one device. This build uses the open source TrustNetZ distribution, from which the entire build can be replicated easily. The TrustNetZ distribution includes source code for the IoT device, the router, the access point, the network onboarding component, the policy engine, the manufacturer services, the registrar and a demo application server. TrustNetZ makes use of NquiringMinds tdx Volt to issue and validate verifiable credentials.
- The BRSKI factory provisioning build is deployed in the Build 5 environment. The IoT device in this build is a Raspberry Pi equipped with an Infineon Optiga SLB 9670 TPM 2.0, which gets provisioned with birth credentials (i.e., a public/private key pair and an IDevID). The BRSKI factory provisioning build also uses an external certificate authority hosted on the premises of NquiringMinds to provide the device certificate signing service.
- The Wi-Fi Easy Connect factory provisioning build is deployed in the Build 1 environment. Its IoT devices are Raspberry Pis equipped with a SEALSQ VaultIC Secure Element, which gets provisioned with a DPP URI. The Secure Element can also be provisioned with an IDevID certificate signed by the SEALSQ INeS certification authority, which is independent of the DPP URI. Code for performing the factory provisioning is stored on an SD card.

Information regarding the physical architecture of all builds, their related collaborators' cloud components, and the shared environment, as well as the baseline software running on these physical architectures, are described in the subsections below. Table 5-1 summarizes the builds that were implemented and provides links to the appendices where each is described in detail.

Table 5-1 Build 1 Products and Technologies

Build	Network-Layer Protocols	Build Champions	Link to Details
Onboarding Builds			
Build 1	Wi-Fi Easy Connect	Aruba/HPE	Appendix C
Build 2	Wi-Fi Easy Connect	CableLabs and OCF	Appendix D
Build 3	BRSKI	Sandelman Software Works	Appendix E
Build 4	Thread	Silicon Labs and Kudelski IoT	Appendix F
Build 5	BRSKI over Wi-Fi	NquiringMinds	Appendix G
Factory Provisioning Builds			
BRSKI with Build 5	BRSKI over WIFI	SEALSQ and NquiringMinds	Appendix H.3
Wi-Fi Easy Connect with Build 1	Wi-Fi Easy Connect	SEALSQ and Aruba/HPE	Appendix H.4

5.1 Shared Environment

The NCCoE IoT Onboarding laboratory contains a shared environment to host several baseline services in support of the builds. These baseline services supported configuration and integration work in each of the builds and allowed collaborators to work together throughout the build process. This shared environment is contained in its own network segment, with access to/from the rest of the lab environment closely controlled. In addition, each of the systems in the shared environment is hardened with baseline configurations.

5.1.1 Domain Controller

The Domain Controller provides Active Directory and Domain Name System (DNS) services supporting network access and access control in the lab. It runs on Windows Server 2019.

5.1.2 Jumpbox

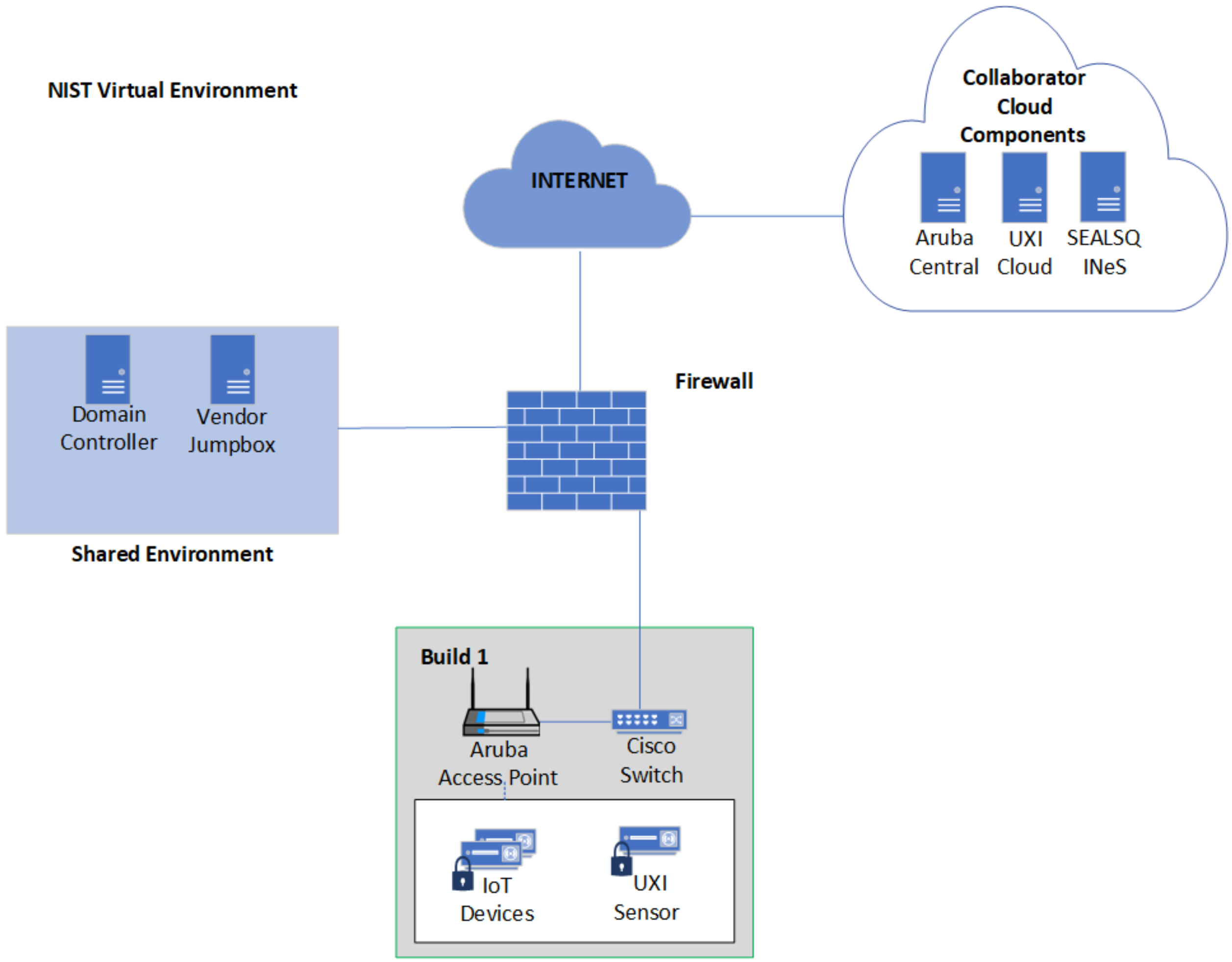
The jumpbox provides secure remote access and management to authorized collaborators on each of the builds. It runs on Windows Server 2019.

5.2 Build 1 (Wi-Fi Easy Connect, Aruba/HPE) Physical Architecture

Figure 5-2 is a view of the high-level physical architecture of Build 1 in the NCCoE IoT Onboarding laboratory. The build components include an Aruba Wireless Access Point, Aruba Central, UXI Cloud, a Cisco Catalyst switch, a SEALSQ INeS CMS CA, and the IoT devices to be onboarded, which include both a Raspberry Pi and a UXI sensor. Most of these components are described in Section 3.4.1 and Section 3.4.3.

- The Aruba Access Point acts as the DPP Configurator and relies on the Aruba Central cloud service for authentication and management purposes.
- Aruba Central ties together the IoT Operations, Client Insights, and Cloud Auth services to support the network-layer onboarding operations of the build. It also provides an API to support the device ownership and bootstrapping information transfer process.
- The Cisco Catalyst Switch provides Power-over-Ethernet and network connectivity to the Aruba Access Point.
- The UXI Sensor acts as an IoT device and onboards to the network via Wi-Fi Easy Connect. After network-layer onboarding, it performs independent (see Section 3.3.2) application-layer onboarding. Once it has application-layer onboarded and is operational on the network, it does passive and active monitoring of applications and services and will report outages, disruptions, and quality of service issues.
- UXI Cloud is an HPE cloud service that the UXI sensor contacts as part of the application-layer onboarding process. The UXI sensor downloads a customer-specific configuration from the UXI Cloud so that the UXI sensor can learn about the customer networks and services it needs to monitor.
- The Raspberry Pi acts as an IoT device and onboards to the network via Wi-Fi Easy Connect.
- SEALSQ Certificate Authority has been integrated with Build 1 to sign network credentials that are issued to IoT devices.

1523 **Figure 5-2 Physical Architecture of Build 1**

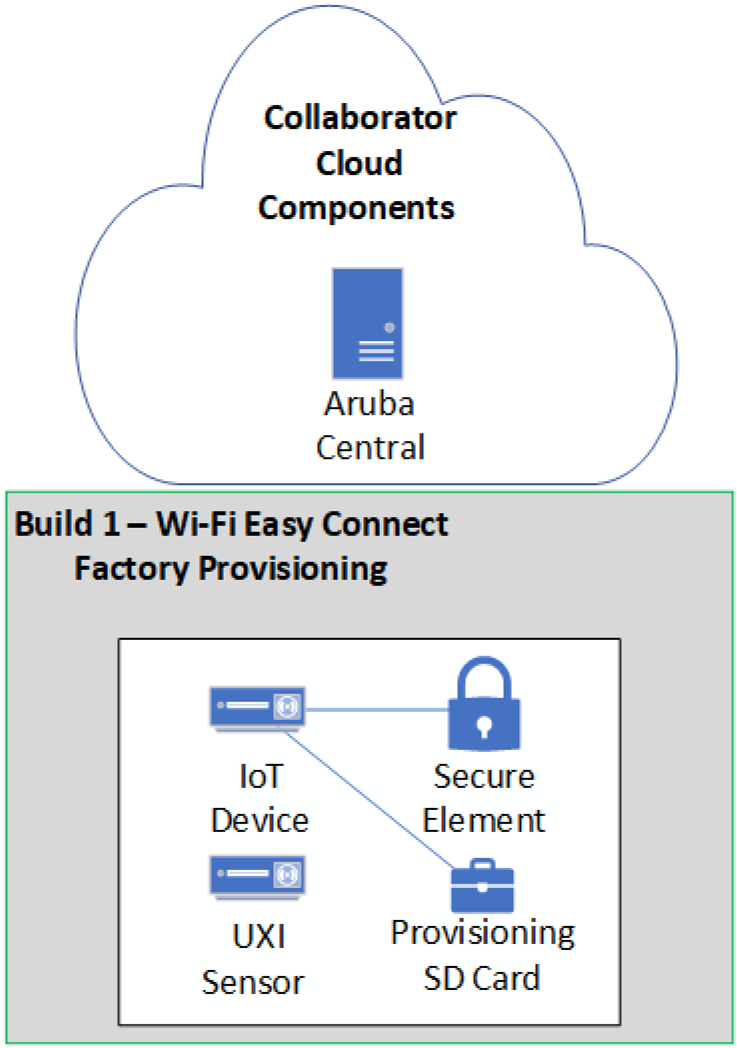


1524 5.2.1 Wi-Fi Easy Connect Factory Provisioning Build Physical Architecture

1525 Figure 5-3 is a view of the high-level physical architecture of the Wi-Fi Easy Connect Factory Provisioning
1526 Build in the NCCoE IoT Onboarding laboratory. The build components include the IoT device, an SD card
1527 with factory provisioning code on it, and a Secure Element. See Appendix H.4 for additional details on
1528 the Wi-Fi Easy Connect Factory Provisioning Build.

1529 - A UXI sensor.
1530 - The IoT Device is a Raspberry Pi.
1531 - The Secure Element is a SEALSQ VaultIC Secure Element and is interfaced with the Raspberry Pi.
1532 The Secure Element both generates and stores the key material necessary to support the DPP
1533 URI during the Factory Provisioning Process.
1534 - An SD card with factory provisioning code.
1535 - Aruba Central provides an API to ingest the DPP URI in support of the device ownership and
1536 bootstrapping information transfer process.

Figure 5-3 Physical Architecture of Wi-Fi Easy Connect Factory Provisioning Build

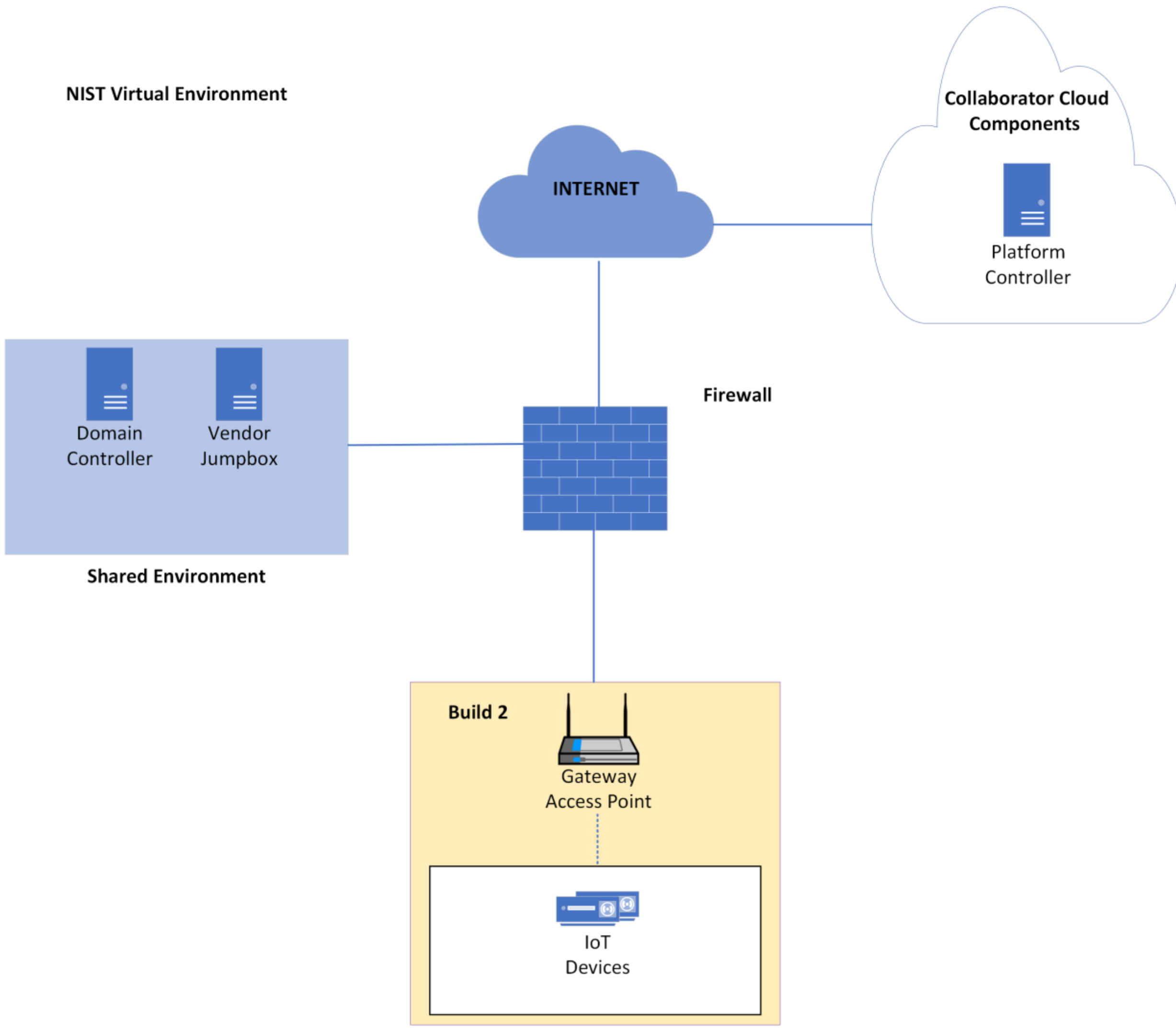


5.3 Build 2 (Wi-Fi Easy Connect, CableLabs, OCF) Physical Architecture

Figure 5-3 is a view of the high-level physical architecture of Build 2 in the NCCoE IoT Onboarding laboratory. The Build 2 components include the Gateway Access Point, three IoT devices, and the Platform Controller, which hosts the application-layer IoTivity service.

- The Gateway Access Point acts as the Custom Connectivity Gateway Agent described in Section 3.4.2.2 and controls all network-layer onboarding activity within the network. It also hosts OCF IoTivity functions, such as the OCF OBT and the OCF Diplomat.
- The Platform Controller described in Section 3.4.2.1 provides management capabilities for the Custom Connectivity Gateway Agent. It also hosts the application-layer IoTivity service for the IoT devices as described in Section 3.4.8.1.
- The IoT devices serve as reference clients, as described in Section 3.4.2.3. They run OCF reference implementations. The IoT devices are onboarded to the network and complete both application-layer and network-layer onboarding.

Figure 5-4 Physical Architecture of Build 2



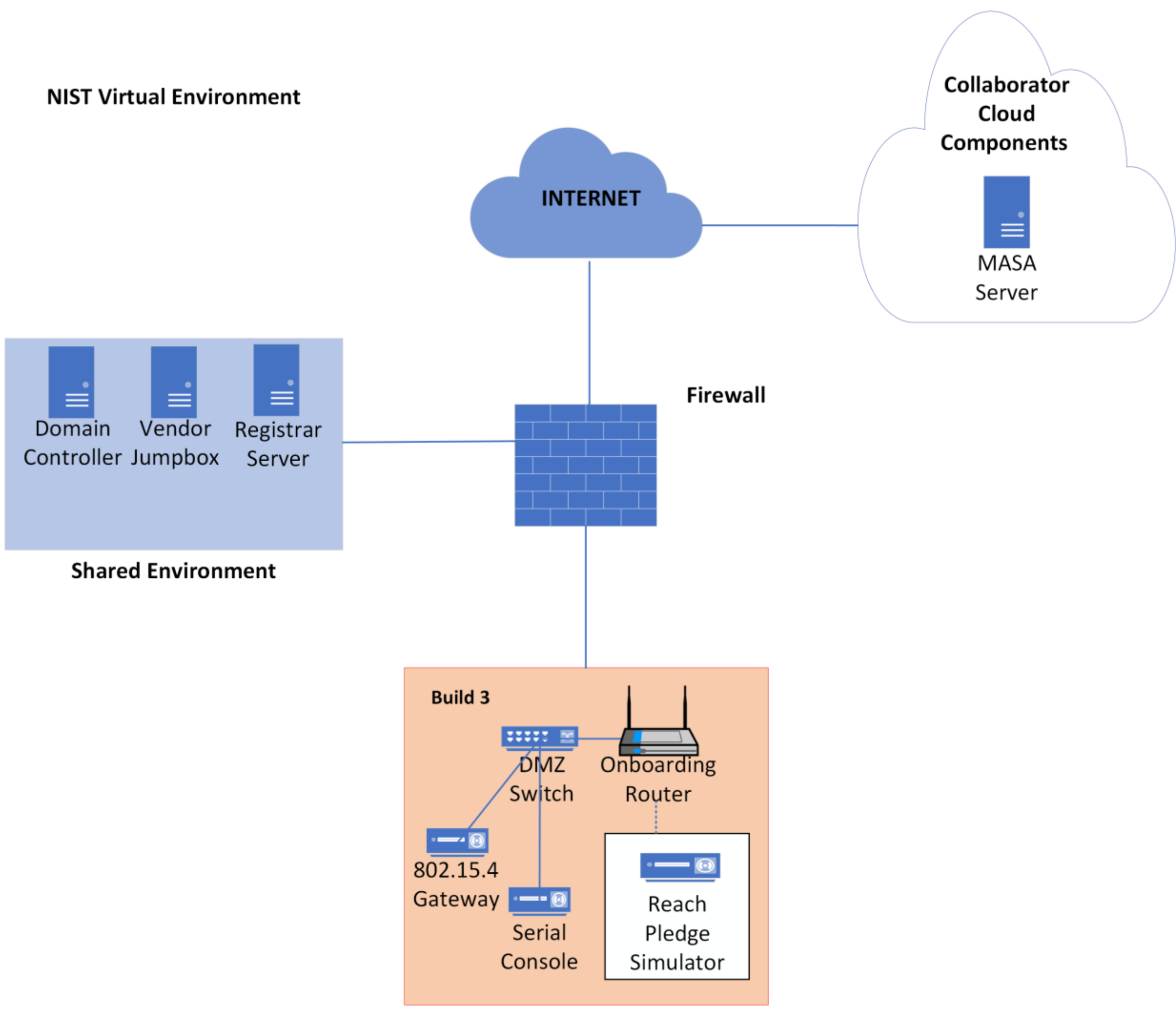
5.4 Build 3 (BRSKI, Sandelman Software Works) Physical Architecture

Figure 5-4 is a view of the high-level physical architecture of Build 3 in the NCCoE IoT Onboarding laboratory. The Build 3 components include the onboarding router, a Registrar Server, a MASA server, a DMZ switch, IoT devices, a serial console, and an 802.15.4 gateway.

- The onboarding router is a Turris MOX router running OpenWRT. The onboarding router quarantines the IoT devices until they complete the BRSKI onboarding process.
- The owner's Registrar Server hosts the Minerva Fountain Join Registrar Coordinator application running in a virtual machine. The Registrar Server determines whether or not a device meets the criteria to join the network.
- The MASA server for this build is a Minerva Highway MASA server as outlined in Section 3.4.9.1. The role of the MASA server is to receive the voucher-request from the Registrar Server and confirm that the Registrar Server has the right to own the device.

- The DMZ switch is a basic Netgear switch that segments the build from the rest of the lab.
- The IoT devices include an ESP32 Xtensa device with Wi-Fi that will be tested with FreeRTOS and RIOT-OS, a Raspberry Pi 3 running Raspbian 11, and an nRF52840 with an 802.15.4 radio that is running RIOT-OS. The IoT devices are currently not used in the build but will serve as clients to be onboarded onto the network in a future implementation of the build.
- The Sandelman Software Works Reach Pledge Simulator is the device that is onboarded to the network in the current build.
- The serial console is a BeagleBone Green with an attached USB hub. The serial console is used to access the IoT devices for diagnostic purposes. It also provides power and power control for USB-powered devices.
- The 802.15.4 gateway is integrated into the Raspberry Pi 3 via an OpenMote daughter card. This gateway will serve to onboard one of the IoT devices in a future implementation of this build.

Figure 5-5 Physical Architecture of Build 3

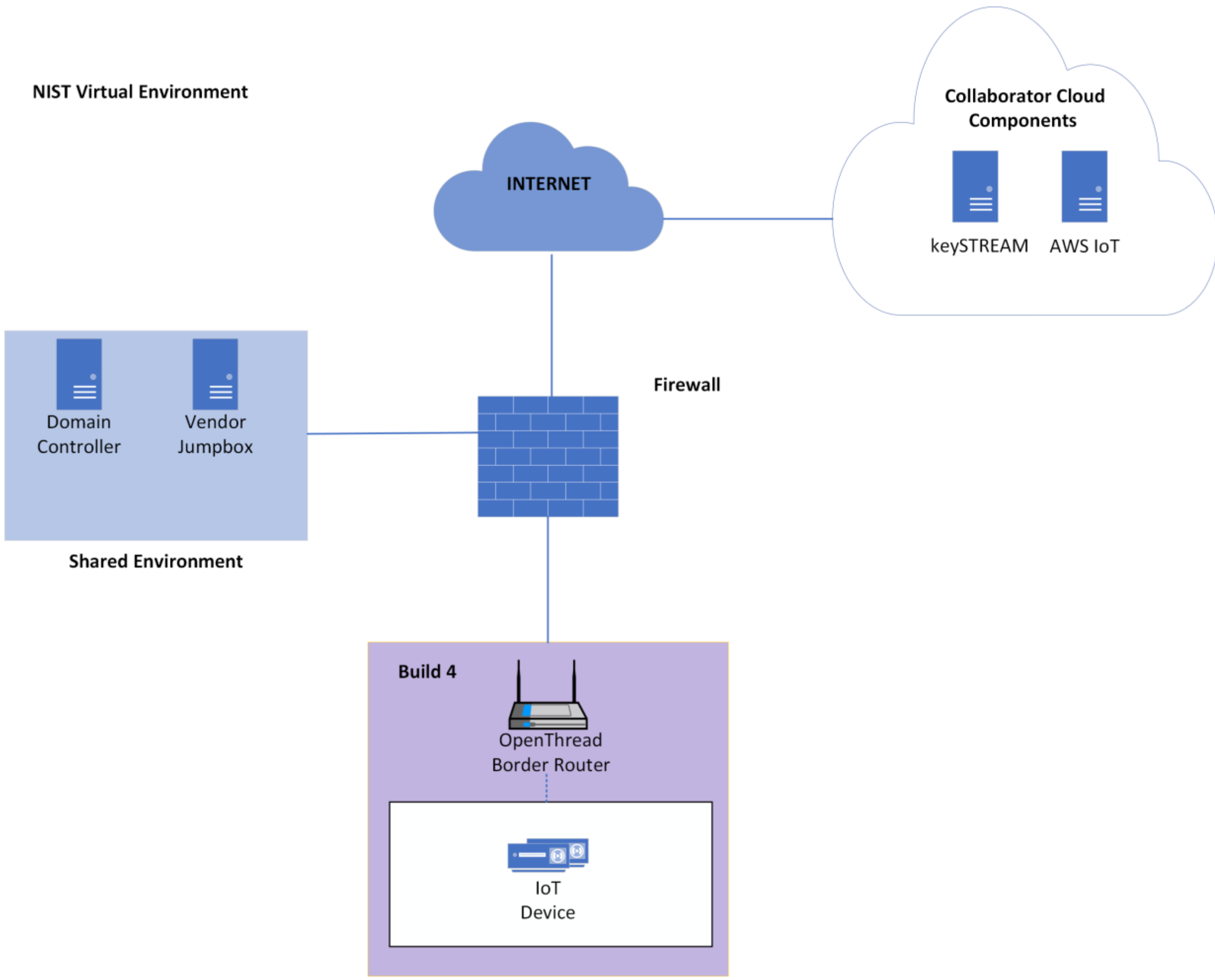


5.5 Build 4 (Thread, Silicon Labs, Kudelski IoT) Physical Architecture

Figure 5-6 is a view of the high-level physical architecture of Build 4 in the NCCoE IoT Onboarding laboratory. The Build 4 components include a keySTREAM server, an AWS IoT server, an OpenThread Border Router, and a Thread IoT device.

- The keySTREAM server described in Section 3.4.5.1 is the application layer onboarding service provided by Kudelski IoT. The IoT device will authenticate to keySTREAM using a Silicon Labs chip birth certificate and private key and leveraging Silicon Labs' Secure Engine in the EFR32MG24 chipset ("Secure Vault(TM) High" which is security certified Platform Security Architecture (PSA)/Security Evaluation Standard for IoT Platforms (SESIP) Level 3 to protect that birth identity with Secure Boot, Secure Debug, and physically unclonable function (PUF) wrapped key storage and hardware tamper protection).
- The AWS IoT server provides the MQTT test client for the trusted application-layer onboarding. The Proof of Possession Certificate is provisioned for the device using a registration code from the AWS server.
- The OpenThread Border Router is run on a Raspberry Pi 3B and serves as the Thread Commissioner and Leader. It communicates with the IoT device by means of a Silicon Labs Gecko Wireless Devkit which serves as the 802.15.4 antenna for the build.
- The IoT Device in this build is a Silicon Labs Thunderboard (BRD2601A) containing the EFR32MG24Bx 15.4 SoC with Secure Vault (TM) High running the Thread protocol. It serves as the child node on the Thread network and is onboarded onto AWS IoT Core using credentials provisioned from the Kudelski keySTREAM service.

Figure 5-6 Physical Architecture of Build 4



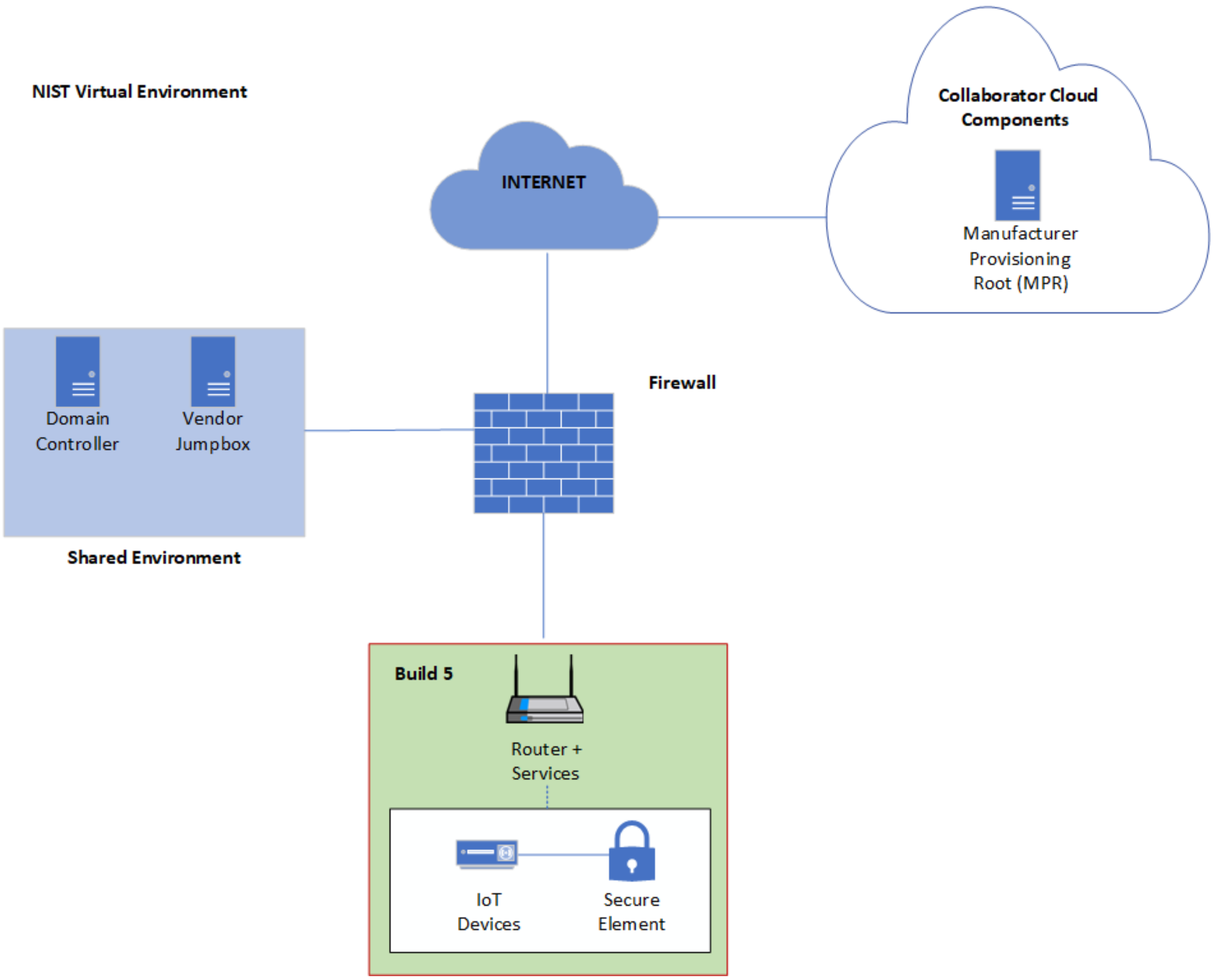
5.6 Build 5 (BRSKI, NquiringMinds) Physical Architecture

Figure 5-6 is a view of the high-level physical architecture of Build 5 in the NCCoE IoT Onboarding laboratory. The Build 5 components include a MASA, Registrar, Router Access Point, an IoT Device, and a Secure Element:

- A Raspberry Pi 4B serves as the MASA, Registrar and Router Access Point for the local network. The role of the MASA is to receive the voucher-request from the Registrar and confirm that the Registrar has the right to own the device. The registrar self-signs credentials, namely the Local Device Identifier (LDevID), issued to the IoT devices. The pledge (IoT device) gets its IDevID certificate for device identity from the Manufacturer Provisioning Root (MPR) server during the factory provisioning process, it can be assumed to be present on the device at the point of onboarding. The Registrar determines whether or not a device meets the criteria to join the network. The router access point runs an open and closed BRSKI network, the closed BRSKI network may only be accessed through secure onboarding, which is performed via the open network. The registrar leverages a local tdx Volt instance to sign and verify verifiable credentials.
- Raspberry Pi 4Bs act as IoT Devices (pledges) for this build.

- The Secure Element is an Infineon Optiga SLB 9670 TPM 2.0 Secure Element, and both generates and stores the key material necessary to support the IDevID certificate during the Factory Provisioning Process, as well as the onboarding process to request the voucher from the MASA via the registrar and the request to the registrar to sign the LDevID. The system can also be configured to use a SEALSQ VaultIC408 secure element. See Appendix H.3 for additional details on the BRSKI factory provisioning builds.

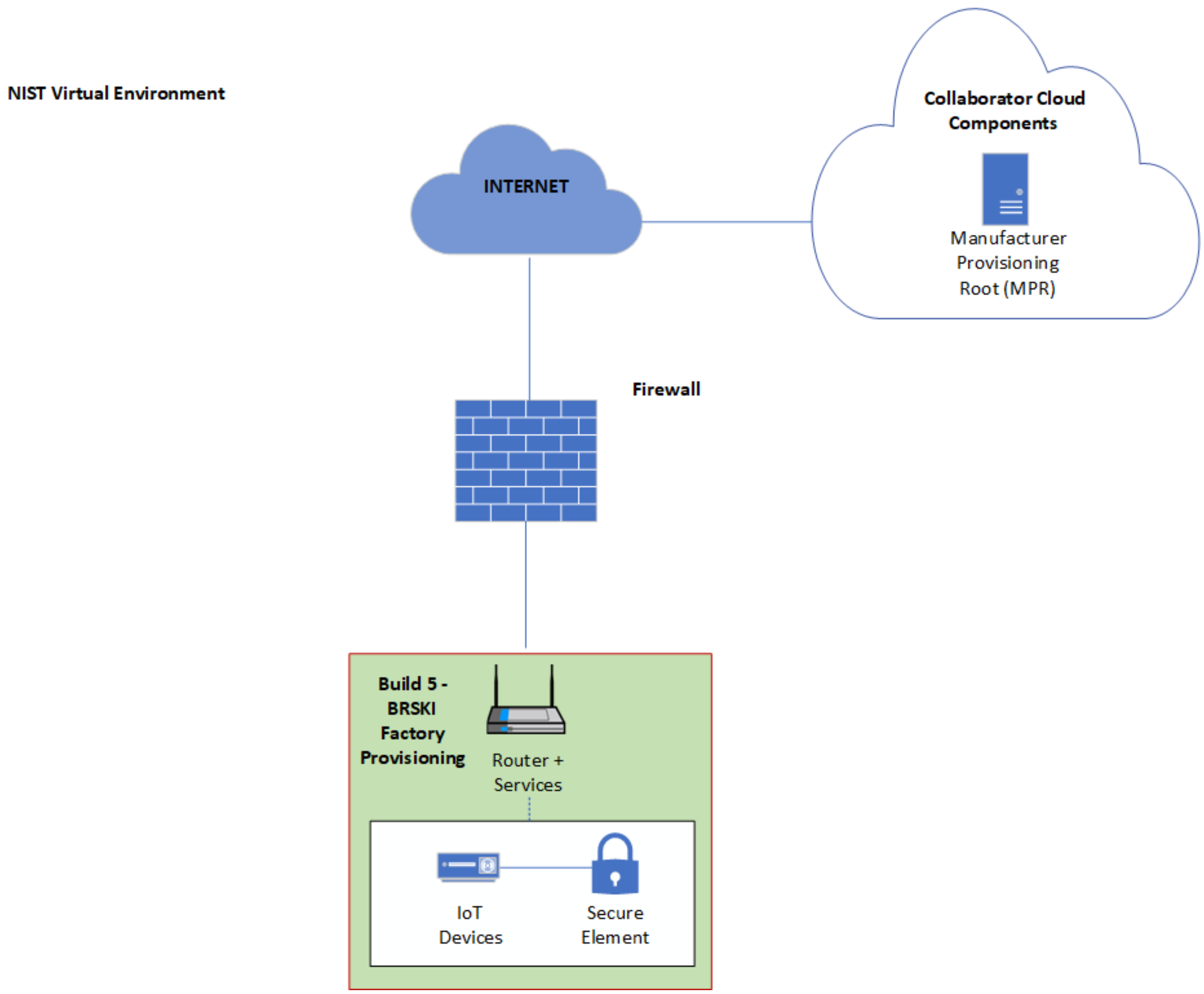
Figure 5-7 Physical Architecture of Build 5



5.6.1 BRSKI Factory Provisioning Build Physical Architecture

Figure 5-8 is a view of the high-level physical architecture of the BRSKI Factory Provisioning Build in the NCCoE IoT Onboarding laboratory. This build uses the same IoT device as Build 5: a Raspberry Pi integrated with an Infineon Optiga SLB 9670 TPM 2.0 Secure Element. The factory provisioning code is hosted on an SD card. When a provisioning event is triggered the IoT device will attempt a connection to a Manufacturer Provisioning Root (MPR) server that sits in the cloud and acts as the certification authority. It signs the IDevID (X.509) certificate, which is then passed back to the IoT device for installation. As in Build 5, the Router + Services hosts a MASA, which is given device identity information in order to verify voucher requests during the BRKSI process. See Appendix H.3 for additional details on the BRSKI factory provisioning builds.

Figure 5-8 Physical Architecture of BRSKI Factory Provisioning Build



6 General Findings

6.1 Wi-Fi Easy Connect

The Wi-Fi Easy Connect solution that was demonstrated in Build 1 and Build 2 supports trusted network-layer onboarding in a manner that is secure, efficient, and flexible enough to meet the needs of various use cases. It is simple enough to be used by consumers, who typically do not have specialized technical knowledge. In addition, to meet the needs of enterprises, it may be used to onboard a large number of devices quickly. Builds 1 and 2 are implementations of this protocol, and they are interoperable: IoT devices that were provisioned for use with Build 1 were able to be onboarded onto the network using Build 2, and IoT devices that were provisioned for use with Build 2 were able to be onboarded onto the network using Build 1.

6.1.1 Mutual Authentication

Although DPP is designed to support authentication of the network by the IoT device as well as authentication of the device by the network, the Wi-Fi Easy Connect solutions that were demonstrated in builds 1 and 2 do not demonstrate mutual authentication at the network layer. They only support authentication of the device. In order to authenticate the network, the device needs to be provided with the DPP URI for the network configurator, which means that the device has to have a functional user interface so that the DPP URI can be input into it. The devices being used in builds 1 and 2 do not have user interfaces.

6.1.2 Mutual Authorization

When using DPP, device authorization is based on possession of the device's DPP URI. When the device is acquired, its DPP URI is provided to the device owner. A trusted administrator of the owner's network is assumed to approve addition of the device's DPP URI to the database or cloud service where the DPP URIs of authorized devices are stored. During the onboarding process, the fact that the owning network is in possession of the device's DPP URI indicates to the network that the device is authorized to join it.

DPP supports network authorization using the Resurrecting Duckling security model [13]. Although the device cannot cryptographically verify that the network is authorized to onboard it, the fact that the network possesses the device's public key is understood by the device to implicitly authorize the network to onboard the device. The assumption is that an unauthorized network would not have possession of the device and so would not be able to obtain the device's public key. While this assurance of authorization is not cryptographic, it does provide some level of assurance that the "wrong" network won't onboard it.

6.1.3 Secure Storage

The UXI sensor used in Build 1 has a TPM where the device's birth credential and private key are stored, providing a secure root of trust. However, the lack of secure storage on some of the other IoT devices (e.g., the Raspberry Pis) used to demonstrate onboarding in Build 2 is a current weakness. Ensuring that the confidentiality of a device's birth, network, and other credentials is protected while stored on the device is an essential aspect of ensuring the security of the network-layer onboarding process, the device, and the network itself. To fully demonstrate trusted network-layer onboarding, devices with secure storage should be used in the future whenever possible.

6.2 BRSKI

The BRSKI solution that is demonstrated in Build 3 supports trusted network-layer onboarding in a manner that is secure, efficient, and able to meet the needs of enterprises. It may be used to onboard a large number of devices quickly onto a wired network. This BRSKI build is based on IETF RFC 8995 [7]. The build has a reliance on the manufacturer to provision keys for the onboarding device and has a reliance on a cloud-based service for the MASA server. The BRSKI solution that is demonstrated in Build 5 provides similar trusted functionality for onboarding devices onto a Wi-Fi network. This BRSKI build is based on an IETF individual draft describing how to run BRSKI over IEEE 802.11 [10].

6.2.1 Reliance on the Device Manufacturer

Organizations implementing BRSKI (whether wired or over Wi-Fi) should be aware of the reliance that they will have on the IoT device manufacturer in properly and securely provisioning their devices. If keys become compromised, attackers may be able to onboard their own devices to the network, revoke certificates to prevent legitimate devices from being onboarded, or onboard devices belonging to others onto the attacker's network using the attacker's MASA. These concerns are addressed in depth in RFC 8995 section 11.6. If a device manufacturer goes out of business or otherwise shuts down their MASA servers, the onboarding services for their devices will no longer function.

During operation, onboarding services may become temporarily unavailable for a number of reasons. In the case of a DoS attack on the MASA, server maintenance, or other outage on the part of the manufacturer, an organization will not be able to access the MASA. These concerns are addressed in depth in RFC 8995 section 11.1.

6.2.2 Mutual Authentication

BRSKI supports authentication of the IoT device by the network as well as authentication of the network by the IoT device. The Registrar authenticates the device when it receives the IDevID from the device. The MASA confirms that the Registrar is the legitimate owner or authorized onboarder of the device and issues a voucher. The device is able to authenticate the network using the voucher that it receives back from the MASA. This process is explained in depth in RFC 8995 section 11.5.

6.2.3 Mutual Authorization

BRSKI authorization for the IoT device is done via the voucher that is returned to the Registrar from the MASA. The voucher states which network the IoT device is authorized to join. The Registrar determines the level of access the IoT device has to the network.

6.2.4 Secure Storage

Build 5 uses a Secure Element attached to the IoT devices (e.g., Raspberry Pi devices) to store the IDevID after it is generated during the factory provisioning process (see Appendix H.3 for more details), however the LDevID is not stored on the Secure Element after network-layer onboarding is completed. The lack of secure storage on the IoT devices (e.g., the Raspberry Pi devices) used to demonstrate onboarding in Build 3 is a current weakness. Ensuring that the confidentiality of a device's birth, network, and other credentials is protected while stored on the device is an essential aspect of ensuring the security of the network-layer onboarding process, the device, and the network itself. To fully demonstrate trusted network-layer onboarding, devices with secure storage should be used in the future whenever possible.

6.3 Thread

We do not have any findings with respect to trusted network-layer onboarding using the Thread commissioning protocol. Build 4 demonstrated the connection of an IoT device to a Thread network, but not trusted onboarding of the Thread network credentials to the device. In Build 4, a passphrase is generated on the IoT device and then a person is required to enter this passphrase into the OpenThread

Border Router's (OTBR) web interface. This passphrase serves as a pre-shared key that the device uses to join the Thread network. Due to the fact that a person must be privy to this passphrase in order to provide it to the OTBR, this network-layer onboarding process is not considered to be trusted, according to the definition of trusted network-layer onboarding that we provided in Section 1.2.

After connecting to the Thread network using the passphrase, the Build 4 device was successfully able to gain access to the public IP network via a border router. This enabled the IoT device that was communicating using the Thread wireless protocol to communicate with cloud services and use them to successfully perform trusted application-layer onboarding to the AWS IoT Core.

6.4 Application-Layer Onboarding

We successfully demonstrated both:

- streamlined application-layer onboarding (to the OCF security domain in Build 2) and
- independent application-layer onboarding (to the UXI cloud in Build 1 and to the AWS IoT Core using the Kudelski keySTREAM service in Build 4).

6.4.1 Independent Application-Layer Onboarding

Support for independent application-layer onboarding requires the device manufacturer to pre-provision the device with software to support application-layer onboarding to the specific application service (e.g., the UXI cloud or the AWS IoT Core) desired. The Kudelski keySTREAM service supports the application-layer onboarding provided in Build 4. KeySTREAM is a device security management service that runs as a SaaS platform on the Amazon cloud. Build 4 relies on an integration that has been performed between Silicon Labs and Kudelski keySTREAM. KeySTREAM has integrated software libraries with the Silicon Lab EFR32MG24 (MG24) IoT device's secure vault to enable the private signing key that is associated with an application-layer certificate to be stored into the secure vault using security controls that are available on the MG24. This integration ensures that application-layer credentials can be provisioned into the vault securely such that no key material is misused or exposed.

Because the device is prepared for application-layer onboarding on behalf of a specific, pre-defined customer in Build 4 and this ownership information is sealed into device firmware, the device is permanently identified as being owned by that customer.

6.4.2 Streamline Application-Layer Onboarding

Support for streamlined application-layer onboarding does not necessarily present such a burden on the device manufacturer to provision application-layer onboarding software and/or credentials to the device at manufacturing time. If desired, the manufacturer could pre-install application-layer bootstrapping information onto the device at manufacturing time, as must be done in the independent application-layer onboarding case. Alternatively, the device manufacturer may simply ensure that the device has the capability to generate one-time application-layer bootstrapping information at runtime and use the secure exchanges inherent in trusted network-layer onboarding to support application-layer onboarding.

7 Additional Build Considerations

The Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management project is now complete, so no additions or changes to the existing builds are planned as part of this project effort. As trusted network-layer onboarding is increasingly adopted, however, others may wish to continue implementation efforts to develop new build capabilities or enhance existing ones, so it is worth noting potential areas of further work. Various ways in which individual builds could be enhanced are noted in the appendices that detail each build's technologies and architectures. For example, some builds could be enhanced by the addition of architectural components that they have not yet implemented, such as secure device storage; the use of an independent, third-party certificate signing authority; support for network-layer onboarding using Thread MeshCoP; support for application-layer onboarding; and support (or enhanced support) for ongoing device authorization. In addition to adding components to support these capabilities, future work could potentially involve demonstration of application-layer onboarding using the FIDO Alliance's FIDO Device Onboard (FDO) specification and/or the Connectivity Standards Alliance (CSA) MATTER specification. Other future work could involve integrating additional security mechanisms with network-layer onboarding, beginning at device boot-up and extending through all phases of the device lifecycle, to further protect the device and, by extension, the network. For example, future builds could include the capability to demonstrate the integration of trusted network-layer onboarding with zero trust-inspired capabilities such as those described in the following subsections. In addition, the scope of implementation efforts could potentially be expanded beyond the current focus on IP-based networks. While this project's goal has been to tackle what is currently implementable, the subsections that follow briefly discuss areas that could potentially be addressed by others in the future.

7.1 Network Authentication

Future builds could be designed to demonstrate network authentication in addition to device authentication as part of the network-layer onboarding process. Network authentication enables the device to verify the identity of the network that will be taking control of it prior to permitting itself to be onboarded.

7.2 Device Communications Intent

Future builds could be designed to demonstrate the use of network-layer onboarding protocols to securely transmit device communications intent information from the device to the network (i.e., to transmit this information in encrypted form with integrity protections). Secure conveyance of device communications intent information, combined with enforcement of it, would enable the build to ensure that IoT devices are constrained to sending and receiving only those communications that are explicitly required for each device to fulfill its purpose. Build 5 currently enforces device communications intent as part of its continuous assurance process. Build 5 determines device communications intent information (e.g., the device's MUD file URL) based on device type rather than conveying this information from the device to the network during onboarding.

7.3 Network Segmentation

Future builds could demonstrate the ability of the onboarding network to dynamically assign each new device that is permitted to join the network to a specific subnetwork. The router may have multiple network segments configured to which an onboarded device may be dynamically assigned. The decision regarding which segment (subnetwork) to which to assign the device could potentially be based on the device's DHCP fingerprint, other markers of the device's type, or some indication of the device's trustworthiness, subject to organizational policy.

7.4 Integration with a Lifecycle Management Service

Future builds could demonstrate trusted network-layer onboarding of a device, followed by streamlined trusted application-layer onboarding of that device to a lifecycle management application service. Such a capability would ensure that, once connected to the local network, the IoT device would automatically and securely establish an association with a trusted lifecycle management service that is designed to keep the device updated and patched on an ongoing basis.

7.5 Network Credential Renewal

Some devices may be provisioned with network credentials that are X.509 certificates and that will, therefore, eventually expire. Future build efforts could explore and demonstrate potential ways of renewing such credentials without having to reprovision the credentials to the devices.

7.6 Integration with Supply Chain Management Tools

Future work could include definition of an open, scalable supply chain integration service that can provide additional assurance of device provenance and trustworthiness automatically as part of the onboarding process. The supply chain integration service could be integrated with the authorization service to ensure that only devices whose provenance meets specific criteria and that reach a threshold level of trustworthiness will be onboarded or authorized.

7.7 Attestation

Future builds could integrate device attestation capabilities with network-layer onboarding to ensure that only IoT devices that meet specific attestation criteria are permitted to be onboarded. In addition to considering the attestation of each device as a whole, future attestation work could also focus on attestation of individual device components, so that detailed attestation could be performed for each board, integrated circuit, and software program that comprises a device.

7.8 Mutual Attestation

Future builds could implement mutual attestation of the device and its application services. In one direction, device attestation could be used to enable a high-value application service to determine whether a device should be given permission to access it. In the other direction, attestation of the application service could be used to enable the device to determine whether it should give the application service permission to access and update the device.

7.9 Behavioral Analysis

Future builds could integrate artificial intelligence (AI) and machine learning (ML)-based tools that are designed to analyze device behavior to spot anomalies or other potential signs of compromise. Any device that is flagged as a potential threat by these tools could have its network credentials invalidated to effectively evict it from the network, be quarantined, or have its interaction with other devices restricted in some way.

7.10 Device Trustworthiness Scale

Future efforts could incorporate the concept of a device trustworthiness scale in which information regarding device capabilities, secure firmware updates, the existence (or not) of a secure element for private key protection, type and version of each of the software components that comprise the device, etc., would be used as input parameters to calculate each device's trustworthiness value. Calculating such a value would essentially provide the equivalent of a background check. A history for the device could be maintained, including information about whether it has ever been compromised, if it has a known vulnerability, etc. Such a trustworthiness value could be provided as an onboarding token or integrated into the authorization service so permission to onboard to the network, or to access certain resources once joined, could be granted or denied based on historical data and trustworthiness measures.

7.11 Resource Constrained Systems

At present, onboarding solutions for technologies such as Zigbee, Z-Wave, and BLE use their own proprietary mechanisms or depend on gateways. In the future, efforts could be expanded to include onboarding in highly resource-constrained systems and non-IP systems without using gateways. Future work could include trying to perform trusted onboarding in these smaller microcontroller-constrained spaces in a standardized way with the goal of bringing more commonality across various solutions without having to rely on IP gateways.

1848
Appendix A List of Acronyms

AAA	Authentication, Authorization, and Accounting
ACL	Access Control List
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AP	Access Point
API	Application Programming Interface
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
BRSKI	Bootstrapping Remote Secure Key Infrastructure
BSS	Basic Service Set
CA	Certificate Authority
CAS	Continuous Authorization Service
CMS	Certificate Management System
CPU	Central Processing Unit
CRADA	Cooperative Research and Development Agreement
CRL	Certificate Revocation List
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarized Zone
DNS	Domain Name System
DPP	Device Provisioning Protocol
DTLS	Datagram Transport Layer Security
ECC	Elliptic Curve Cryptography
ESP	(Aruba) Edge Services Platform
ESS	Extended Service Set
EST	Enrollment over Secure Transport
HPE	Hewlett Packard Enterprise
HSM	Hardware Security Module
HTTPS	Hypertext Transfer Protocol Secure

IDevID	Initial Device Identifier
IE	Information Element
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
LAN	Local Area Network, Local Area Networking
LDevID	Local Device Identifier
LmP	Linux microPlatform
MASA	Manufacturer Authorized Signing Authority
MeshCoP	Thread Mesh Commissioning Protocol
ML	Machine Learning
mPKI	Managed Public Key Infrastructure
MUD	Manufacturer Usage Description
NAC	Network Access Control
NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
OBT	Onboarding Tool
OCF	Open Connectivity Foundation
OCSP	Online Certificate Status Protocol
OS	Operating System
OTA	Over the Air
OTBR	OpenThread Border Router
PKI	Public Key Infrastructure
PSK	Pre-Shared Key
R&D	Research & Development
RBAC	Role-Based Access Control

RCP	Radio Coprocessor
RESTful	Representational State Transfer
RFC	Request for Comments
RoT	Root of Trust
RSA	Rivest-Shamir-Adleman (public-key cryptosystem)
SaaS	Software as a Service
SE	Secure Element
SEF	Secure Element Factory
SoC	System-on-Chip
SP	Special Publication
SSID	Service Set Identifier
SSW	Sandelman Software Works
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOFU	Trust On First Use
TPM	Trusted Platform Module
URI	Uniform Resource Identifier
UXI	(Aruba) User Experience Insight
VM	Virtual Machine
WAN	Wide Area Network, Wide Area Networking
WFA	Wi-Fi Alliance
WPA2	Wi-Fi Protected Access 2
WPA3	Wi-Fi Protected Access 3

1849
Appendix B Glossary

Application-Layer Bootstrapping Information	Information that the device and an application-layer service must have in order for them to mutually authenticate and use a trusted application-layer onboarding protocol to onboard a device at the application layer. There is application-layer bootstrapping information about the device that the network must be in possession of, and application-layer bootstrapping information about the application service that the device must be in possession of. A typical example of application-layer bootstrapping information that the device must have is the public key that corresponds to the trusted application service's private key.
Application-Layer Onboarding	The process of providing IoT devices with the application-layer credentials they need to establish a secure (i.e., encrypted) association with a trusted application service. This document defines two types of application-layer onboarding: independent and streamlined.
Independent Application-Layer Onboarding	An application-layer onboarding process that does not rely on use of the network-layer onboarding process to transfer application-layer bootstrapping information between the device and the application service.
Network-Layer Bootstrapping Information	Information that the device and the network must have in order for them to use a trusted network-layer onboarding protocol to onboard a device. There is network-layer bootstrapping information about the device that the network must be in possession of, and network-layer bootstrapping information about the network that the device must be in possession of. A typical example of device bootstrapping information that the network must have is the public key that corresponds with the device's private key.
Network-Layer Onboarding	The process of providing IoT devices with the network-layer credentials and policy they need to join a network upon deployment.
Streamlined Application-Layer Onboarding	An application-layer onboarding process that uses the network-layer onboarding protocol to securely transfer application-layer bootstrapping information between the device and the application service.

Trusted Network-Layer Onboarding	A network-layer onboarding process that meets the following criteria:
	• provides each device with unique network credentials,
	• enables the device and the network to mutually authenticate,
	• sends devices their network credentials over an encrypted channel,
	• does not provide any person with access to the network credentials, and
	• can be performed repeatedly throughout the device lifecycle to enable:
	    • the device's network credentials to be securely managed and replaced as needed, and
	    • the device to be securely onboarded to other networks after being repurposed or resold.

Appendix C Build 1 (Wi-Fi Easy Connect, Aruba/HPE)

C.1 Technologies

Build 1 is an implementation of network-layer onboarding that uses the Wi-Fi Easy Connect protocol. The onboarding infrastructure and related technology components for Build 1 have been provided by Aruba/HPE. IoT devices that were onboarded using Build 1 were provided by Aruba/HPE and CableLabs. The CA used for signing credentials issued to IoT devices was provided by SEALSQ, a subsidiary of WISeKey. For more information on these collaborators and the products and technologies that they contributed to this project overall, see Section 3.4.

Build 1 network onboarding infrastructure components within the NCCoE lab consist of the Aruba Access Point. Build 1 also requires support from Aruba Central and the UXI Cloud, which are accessed via the internet. IoT devices that can be network-layer onboarded using Build 1 include the Aruba/HPE UXI sensor and CableLabs Raspberry Pi. The UXI sensor also includes the Aruba UXI Application, which enables it to use independent (see Section 3.3.2) application-layer onboarding to be onboarded at the application layer as well, providing that the network to which the UXI sensor is onboarded has connectivity to the UXI Cloud via the internet. The Build 1 implementation supports the provisioning of all three types of network credentials defined in DPP:

- Connector for DPP-based network access
- Password/passphrase/PSK for WPA3/WPA2 network access
- X.509 certificates for 802.1X network access

Build 1 has been integrated with the SEALSQ CA on SEALSQ INeS CMS to enable Build 1 to obtain signed certificates from this CA when Build 1 is onboarding devices and issuing credentials for 802.1X network access. When issuing credentials for DPP and WPA3/WPA2-based network access, the configurator does not need to use a CA.

Table C-1 lists the technologies used in Build 1. It lists the products used to instantiate each component of the reference architecture and describes the security function that the component provides. The components listed are logical. They may be combined in physical form, e.g., a single piece of hardware may house a network onboarding component, a router, and a wireless access point.

Table C-1 Build 1 Products and Technologies

Component	Product	Function
Network-Layer Onboarding Component (Wi-Fi Easy Connect Configurator)	Aruba Access Point with support from Aruba Central	Runs the Wi-Fi Easy Connect network-layer onboarding protocol to interact with the IoT device to perform one-way or mutual authentication, establish a secure channel, and securely provide local network credentials to the device. If the network credential that is being provided to the device is a certificate, the onboarding component will interact with a certificate authority to sign the certificate. The configurator deployed in Build 1 supports DPP 2.0, but it is also backward compatible with DPP 1.0.

Component	Product	Function
Access Point, Router, or Switch	Aruba Access Point	Wireless access point that also serves as a router. It may get configured with per-device access control lists (ACLs) and policy when devices are onboarded.
Supply Chain Integration Service	Aruba Central	The device manufacturer provides device bootstrapping information to the HPE Cloud via the REST API that is documented in the DPP specification. Once the device is transferred to an owner, the HPE Cloud provides the device bootstrapping information (i.e., the device's DPP URI) to the device owner's private tenancy within the HPE Cloud.
Authorization Service	Cloud Auth (on Aruba Central)	The authorization service provides the configurator and router with the information needed to determine if the device is authorized to be onboarded to the network and, if so, whether it should be assigned any special roles or be subject to any specific access controls. It provides device authorization, role-based access control, and policy enforcement.
Build-Specific IoT Device	Aruba UXI Sensor	The IoT device that is used to demonstrate both trusted network-layer onboarding and trusted application-layer onboarding. It runs the Wi-Fi Easy Connect network-layer onboarding protocol supported by the build to securely receive its network credentials. It also has an application that enables it to perform independent (see Section 3.3.2) application-layer onboarding.
Generic IoT Device	Raspberry Pi	The IoT device that is used to demonstrate only trusted network-layer onboarding.
Secure Storage	Aruba UXI Sensor Trusted Platform Module (TPM)	Storage on the IoT device that is designed to be protected from unauthorized access and capable of detecting attempts to hack or modify its contents. Used to store and process private keys, credentials, and other information that must be kept confidential.
Certificate Authority (CA)	SEALSQ INeS CMS CA	Issues and signs certificates as needed. These certificates can be used by the device to connect to any 802.1a-based network.
Application-Layer Onboarding Service	UXI Application and UXI Cloud	After connecting to the network, the device downloads its application-layer credentials from the UXI cloud and uses them to authenticate to the UXI application, with which it interacts.

Component	Product	Function
Ongoing Device Authorization	N/A – Not intended for inclusion in this build	Performs activities designed to provide an ongoing assessment of the device's trustworthiness and authorization to access network resources. For example, it may perform behavioral analysis or device attestation and use the results to determine whether the device should be granted access to certain high-value resources, assigned to a particular network segment, or other action taken.
Manufacturer Factory Provisioning Process	N/A (Not implemented at the time of publication)	Manufactures the IoT device. Creates, signs, and installs the device's unique identity and other birth credentials into secure storage. Installs information the device requires for application-layer onboarding (if applicable). May populate a manufacturer database with information regarding devices that are created and, when the devices are sold, may record what entity owns them.

C.2 Build 1 Architecture

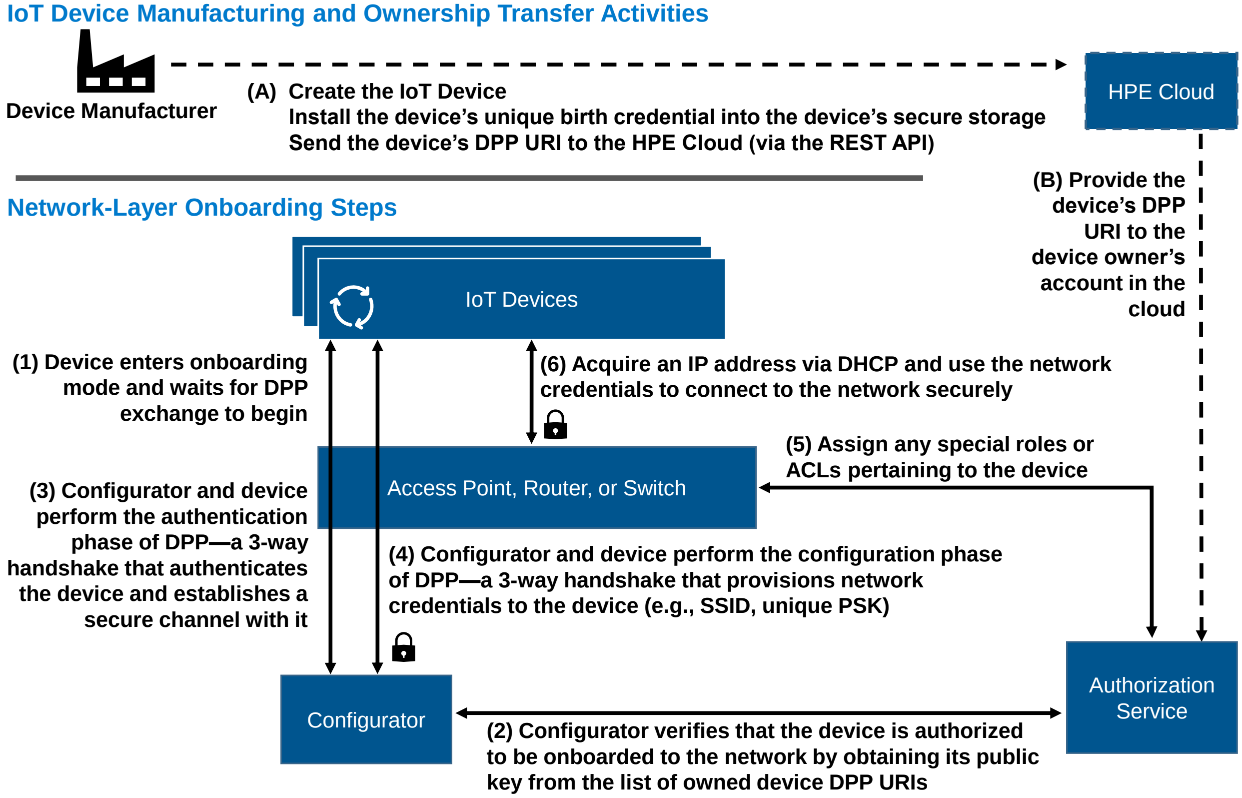
C.2.1 Build 1 Logical Architecture

The network-layer onboarding steps that are performed in Build 1 are depicted in Figure C-1. These steps are broken into two main parts: those required to transfer device bootstrapping information from the device manufacturer to the device owner's authorization service (labeled with letters) and those required to perform network-layer onboarding of the device (labeled with numbers).

The device manufacturer:

1. Creates the device and installs a unique birth credential into secure storage on the device. Then the manufacturer sends the device's bootstrapping information, which takes the form of a DPP URI, to Aruba Central in the HPE cloud. The device manufacturer interfaces with the HPE cloud via a REST API.
2. When the device is purchased, the device's DPP URI is sent to the HPE cloud account of the device's owner. The device owner's cloud account contains the DPP URIs for all devices that it owns.

Figure C-1 Logical Architecture of Build 1



After obtaining the device, the device owner provisions the device with its network credentials by performing the following network-layer onboarding steps:

1. The owner puts the device into onboarding mode. The device waits for the DPP exchange to begin. This exchange includes the device issuing a discovery message, which the owner's configurator hears. The discovery message is secured such that it can only be decoded by an entity that possesses the device's DPP URI.
2. The configurator consults the list of DPP URIs of all owned devices to decode the discovery message and verify that the device is owned by the network owner and is therefore assumed to be authorized to be onboarded to the network.
3. Assuming the configurator finds the device's DPP URI, the configurator and the device perform the authentication phase of DPP, which is a three-way handshake that authenticates the device and establishes a secure (encrypted) channel with it.
4. The configurator and the device use this secure channel to perform the configuration phase of DPP, which is a three-way handshake that provisions network credentials to the device, along with any other information that may be needed, such as the network SSID.
5. The router or switch consults the owner's authentication, authorization, and accounting (AAA) service to determine if the device should be assigned any special roles or if any special ACL entries should be made for the device. If so, these are configured on the router or switch.

6. The device uses Dynamic Host Configuration Protocol (DHCP) to acquire an IP address and then uses its newly provisioned network credentials to connect to the network securely.

This completes the network-layer onboarding process.

After the device is network-layer onboarded and connects to the network, it automatically performs independent (see Section 3.3.2) application-layer onboarding. The application-layer onboarding steps are not depicted in Figure C-1. During the application-layer onboarding process, the IoT device, which is a UXI sensor, authenticates itself to the UXI cloud using its manufacturing certificate and pulls its application-layer credentials from the UXI cloud. In addition, if a firmware update is relevant, this also happens. The UXI sensor contacts the UXI cloud service to download a customer-specific configuration that tells it what to monitor on the customer's network. The UXI sensor then conducts the network performance monitoring functions it is designed to perform and uploads the data it collects to the UXI application dashboard.

C.2.2 Build 1 Physical Architecture

Section 5.2 describes the physical architecture of Build 1.

Appendix D Build 2 (Wi-Fi Easy Connect, CableLabs, OCF)

D.1 Technologies

Build 2 is an implementation of network-layer onboarding that uses the Wi-Fi Easy Connect protocol. Build 2 also supports streamlined (see Section 3.3.2) application-layer onboarding to the OCF security domain. The network-layer onboarding infrastructure for Build 2 is provided by CableLabs and the application-layer onboarding infrastructure is provided by OCF. IoT devices that were network-layer onboarded using Build 2 were provided by Aruba/HPE and OCF. Only the IoT devices provided by OCF were capable of being both network-layer onboarded and streamlined application-layer onboarded. For more information on these collaborators and the products and technologies that they contributed to this project overall, see Section 3.4.

Build 2 onboarding infrastructure components consist of the CableLabs Custom Connectivity Gateway Agent, which runs on the Gateway Access Point, and the Platform Controller. IoT devices onboarded by Build 2 include the Aruba UXI Sensor and CableLabs Raspberry Pi.

Table D-1 lists the technologies used in Build 2. It lists the products used to instantiate each logical build component and the security function that the component provides. The components listed are logical. They may be combined in physical form, e.g., a single piece of hardware may house a network onboarding component, a router, and a wireless access point.

Table D-1 Build 2 Products and Technologies

Component	Product	Function
Network-Layer Onboarding Component (Configurator)	CableLabs Custom Connectivity Gateway Agent with support from CableLabs Platform Controller	Runs the Wi-Fi Easy Connect network-layer onboarding protocol to interact with the IoT device to perform one-way or mutual authentication, establish a secure channel, and securely provide local network credentials to the device. It also securely conveys application-layer bootstrapping information to the device as part of the Wi-Fi Easy Connect protocol to support application-layer onboarding. The network-layer onboarding component deployed in Build 2 supports DPP 2.0, but it is also backward compatible with DPP 1.0.
Access Point, Router, or Switch	Raspberry Pi (running Custom Connectivity Gateway Agent)	The access point includes a configurator that runs the Wi-Fi Easy Connect Protocol. It also serves as a router that: 1) routes all traffic exchanged between IoT devices and the rest of the network, and 2) assigns each IoT device to a local network segment appropriate to the device's trust level (optional).

Component	Product	Function
Supply Chain Integration Service	CableLabs Platform Controller/IoTivity Cloud Service	The device manufacturer provides device bootstrapping information (i.e., the DPP URI) to the CableLabs Web Server. There are several potential mechanisms for sending the DPP URI to the CableLabs Web Server. The manufacturer can send the device's DPP URI to the Web Server directly, via an API. The API used is not the REST API that is documented in the DPP specification. However, the API is published and was made available to manufacturers wanting to onboard their IoT devices using Build 2. Once the device is transferred to an owner, the CableLabs Web Server provides the device's DPP URI to the device owner's authorization service, which is part of the owner's configurator.
Authorization Service	CableLabs Platform Controller	The authorization service provides the configurator and router with the information needed to determine if the device is authorized to be onboarded to the network and, if so, whether it should be assigned any special roles, assigned to any specific network segments, or be subject to any specific access controls.
Build-Specific IoT Device	Raspberry Pi (Bulb) Raspberry Pi (switch)	The IoT devices that are used to demonstrate both trusted network-layer onboarding and trusted application-layer onboarding. They run the Wi-Fi Easy Connect network-layer onboarding protocol to securely receive their network credentials. They also support application-layer onboarding of the device to the OCF environment by conveying the device's application-layer bootstrapping information as part of the network-layer onboarding protocol.
Generic IoT Device	Aruba UXI Sensor	The IoT device that is used to demonstrate only trusted network-layer onboarding.
Secure Storage	N/A (IoT device is not equipped with secure storage)	Storage designed to be protected from unauthorized access and capable of detecting attempts to hack or modify its contents. Used to store and process private keys and other information that must be kept confidential.
Certificate Authority	N/A (Not implemented at the time of publication)	Issues and signs certificates as needed.
Application-Layer Onboarding Service	OCF Diplomat and OCF OBT within IoTivity	After connecting to the network, the OCF Diplomat authenticates the devices, establishes secure channels with them, and sends them access control lists that control which bulbs each switch is authorized to turn on and off.

Component	Product	Function
Ongoing Device Authorization	N/A – Not intended for inclusion in this build	Performs activities designed to provide ongoing assessment of the device's trustworthiness and authorization to access network resources. For example, it may perform behavioral analysis or device attestation and use the results to determine whether the device should be granted access to certain high-value resources, assigned to a particular network segment, or other action taken.
Manufacturer Factory Provisioning Process	N/A (Not yet implemented)	Manufactures the IoT device. Creates, signs, and installs the device's unique identity and other birth credentials into secure storage. Installs information the device requires for application-layer onboarding (if applicable). May populate a manufacturer database with information regarding devices that are created and, when the devices are sold, may record what entity owns them.

D.2 Build 2 Architecture

D.2.1 Build 2 Logical Architecture

The network-layer onboarding steps that are performed in Build 2 are depicted in Figure D-1. These steps are broken into two main parts: those required to transfer device bootstrapping information from the device manufacturer to the device owner's authorization service (labeled with letters) and those required to perform network-layer onboarding of the device (labeled with numbers).

The device manufacturer:

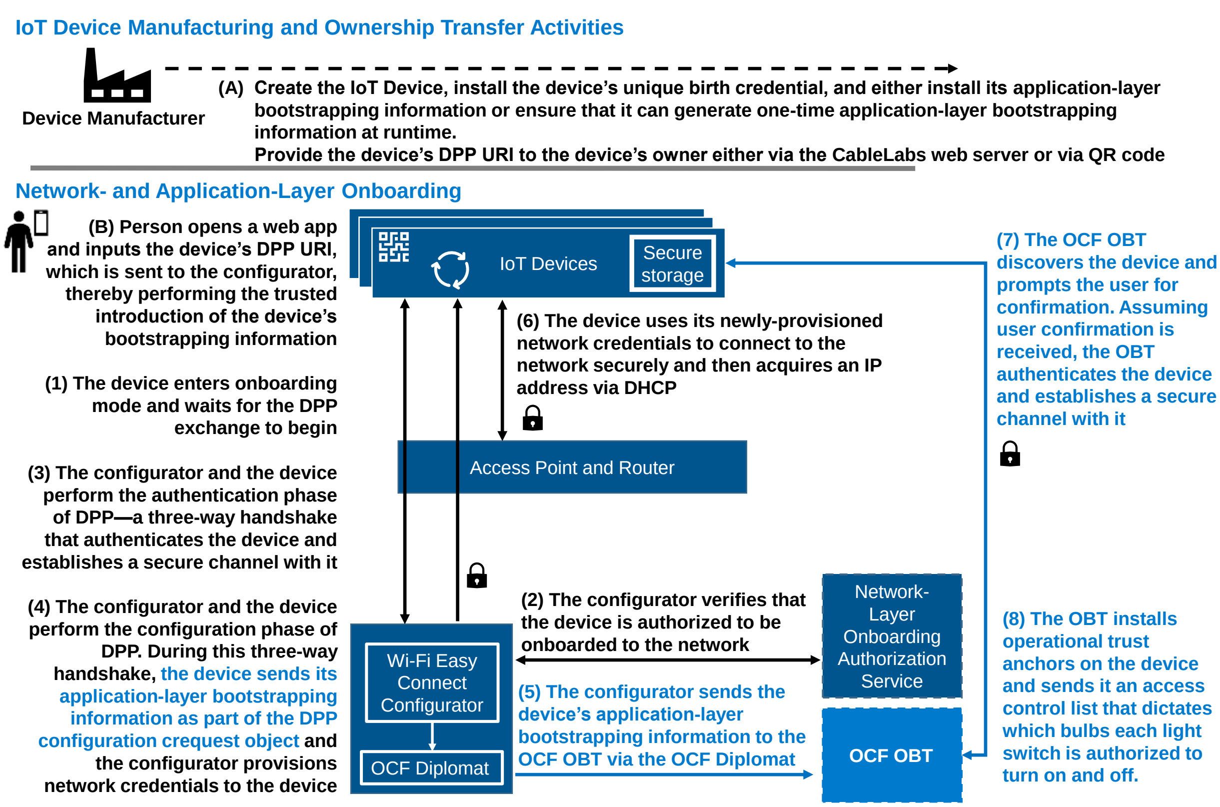
1. Creates the device and installs a unique birth credential into secure storage on the device. Because the device created for use in Build 2 will also perform application-layer onboarding into the OCF security domain, as part of the manufacturing process the manufacturer also either installs application-layer bootstrapping information onto the device or ensures that the device has the capability to generate one-time application-layer bootstrapping information at runtime. Then the manufacturer makes the device's network-layer bootstrapping information, which takes the form of a DPP URI, available to the device's owner.

   Build 2 supports several mechanisms whereby the manufacturer can make the device's network-layer bootstrapping information (i.e., its DPP URI) available to the device owner. The device's DPP URI can be uploaded directly to a device owner's cloud account or web server via API (as might come in handy when onboarding many enterprise devices at one time). Alternatively, the DPP URI can be manually entered into a local web portal that runs a configuration webpage that a device on the same Wi-Fi network can connect to for purposes of scanning a QR code or typing in the DPP URI. A DPP URI that is to be entered manually could, for example, be emailed to the owner or encoded into a QR code and printed on the device chassis, in device documentation, or on device packaging. Table D-1 depicts the case in which the manufacturer provides the device's DPP URI to the owner for manual entry. When the owner receives the device's DPP URI, the owner may optionally add the device's DPP URI to a list of

DPP URIs for devices that it owns that is maintained as part of the owner's authorization service. Such a list would enable the owner's network to determine if a device is authorized to be onboarded to it.

2. The person onboarding the device opens a web application and enters the device's DPP URI. The web application then sends the DPP URI to the Wi-Fi Easy Connect configurator, e.g., through a web request. (Note: Although the laboratory implementation of Build 2 requires the user to enter the DPP URI via a web page, an implementation designed for operational use would typically require the user to provide the DPP URI by scanning a QR code into a network operator-provided app that is logged into the user's account.)

Figure D-1 Logical Architecture of Build 2



After ensuring that the device's network-layer bootstrapping information (i.e., its DPP URI) has been uploaded to the configurator, the device owner performs both trusted network-layer onboarding and streamlined application-layer onboarding to the OCF security domain by performing the steps depicted in Figure D-1. In this diagram, the components that relate to network-layer onboarding are depicted in dark blue and their associated steps are written in black font. The components and steps that are related to application-layer onboarding are depicted in light blue. The steps are as follows:

1. The owner puts the device into onboarding mode. The device waits for the DPP exchange to begin. This exchange includes the device issuing a discovery message, which the owner's configurator hears. The discovery message is secured such that it can only be decoded by an entity that possesses the device's DPP URI.

2. Optionally, if such a list is being maintained, the configurator consults the list of DPP URIs of all owned devices to verify that the device is owned by the network owner and is, therefore, assumed to be authorized to be onboarded to the network. (If the device is being onboarded by an enterprise, the enterprise would likely maintain such a list; however, if the device is being onboarded to a home network, this step might be omitted.)

3. Assuming the configurator finds the device's DPP URI, the configurator and the device perform the authentication phase of DPP, which is a three-way handshake that authenticates the device and establishes a secure (encrypted) channel with it.

4. The configurator and the device use this secure channel to perform the configuration phase of DPP, which is a three-way handshake that provisions network credentials to the device, along with any other information that may be needed, such as the network SSID. In particular, as part of the three-way handshake in the Build 2 demonstration, the device sends its application-layer bootstrapping information to the configurator as part of the DPP configuration request object.

5. The configurator receives the device's application-layer bootstrapping information and forwards it to the OCF Diplomat. The purpose of the OCF Diplomat is to provide a bridge between the network and application layers. It accomplishes this by parsing the org.openconnectivity fields of the DPP request object, which contains the UUID of the device and the application-layer bootstrapping credentials, and sending these to the OCF OBT as part of a notification that the OBT has a new device to onboard. The Diplomat and the OBT use a subscribe and notify mechanism to ensure that the OBT will receive the onboarding request even if the OBT is unreachable for a period of time (e.g., the OBT is out of the home).

6. The device uses its newly provisioned network credentials to connect to the network securely and then uses DHCP to acquire an IP address. This completes the network-layer onboarding process.

7. The OBT implements a filtered discovery mechanism using the UUID provided from the OCF Diplomat to discover the new device on the network. Once it discovers the device, before proceeding, the OBT may optionally prompt the user for confirmation that they want to perform application-layer onboarding to the OCF security domain. This prompting may be accomplished, for example, by sending a confirmation request to an OCF app on the user's mobile device. Assuming the user responds affirmatively, the OBT uses the application-layer bootstrapping information to authenticate the device and take ownership of it by setting up a Datagram Transport Layer Security (DTLS) connection with the device.

8. The OBT then installs operational trust anchors and access control lists onto the device. For example, in the access control list, each light bulb may have an access control entry dictating which light switches are authorized to turn it on and off. This completes the application-layer onboarding process.

Note that, at this time, the application-layer bootstrapping information is provided unilaterally in the Build 2 application-layer onboarding demonstration. The application-layer bootstrapping information of the device is provided to the OCF Diplomat, enabling the OBT to authenticate the device. In a future version of this process, the application-layer bootstrapping information could be provided bidirectionally, meaning that the OCF Diplomat could also send the OCF operational root of trust to the IoT device as part of the DPP configuration response frame. Exchanging application-layer bootstrapping information bilaterally in this way would enable the secure channel set up as part of the network-layer onboarding process to support establishment of a mutually authenticated session between the device and the OBT.

In the Build 2 demonstration, two IoT devices, a switch and a light bulb, are onboarded at both the network and application layers. Each of these devices sends the OCF Diplomat its application-layer bootstrapping information over the secure network-layer onboarding channel during the network-layer onboarding process. Immediately after they complete the network-layer onboarding process and connect to the network, the OCF Diplomat provides their application-layer bootstrapping information to the OBT. The OBT then uses the provided application-layer bootstrapping information to discover, authenticate, and onboard each device. Because the devices have no way to authenticate the identity of the OBT in the current implementation, the devices are configured to trust the OBT upon first use.

After the OBT authenticates the devices, it establishes secure channels with them and provisions them access control lists that control which bulbs each switch is authorized to turn on and off. To demonstrate that the application onboarding was successful, Build 2 demonstrates that the switch is able to control only those bulbs that the OCF OBT has authorized it to.

D.2.2 Build 2 Physical Architecture

Section 5.3 describes the physical architecture of Build 2.

Appendix E Build 3 (BRSKI, Sandelman Software Works)

E.1 Technologies

Build 3 is an implementation of network-layer onboarding that uses the BRSKI protocol. Build 3 does not support application-layer onboarding. The network-layer onboarding infrastructure and related technology components for Build 3 were provided by Sandelman Software Works. The Raspberry Pi, ESP32, and Nordic NRF IoT devices that will be onboarded in a future implementation of Build 3 were also provided by Sandelman Software Works, as was the Sandelman Software Works Reach Pledge Simulator, which is the device that is onboarded in the current build. The IoT devices do not have secure storage, but future plans are to integrate them with secure storage elements. Build 3 issues private PKI certificates as network credentials at this time, but future plans are to integrate Build 3 with a third-party private CA from which it can obtain signed certificates. For more information on Sandelman Software Works and the products and technologies that it contributed to this project overall, see Section 3.4.

Onboarding Build 3 infrastructure components consist of Raspberry Pi, Nordic NRF, ESP32, Sandelman Software Works Minerva Fountain Join Registrar/Coordinator, Sandelman Software Works Minerva. Highway, Sandelman Software Works Reach Pledge Simulator, and a Minerva Fountain internal CA.

Table E-1 lists the technologies used in Build 3. It lists the products used to instantiate each logical build component and the security function that the component provides. The components are logical. They may be combined in physical form, e.g., a single piece of hardware may house both a network onboarding component and a router and/or wireless access point.

Table E-1 Build 3 Products and Technologies

Component	Product	Function
Network-Layer Onboarding Component (BRSKI Domain Registrar)	Sandelman Software Works Minerva Fountain Registrar	Runs the BRSKI protocol. It authenticates the IoT device, receives a voucher-request from the IoT device, and passes the request to the MASA. It also receives a voucher from the MASA, verifies it, and passes it to the IoT device. Assuming the IoT device finds the voucher to be valid and determines that the network is authorized to onboard it, the Domain Registrar provisions network credentials to the IoT device using EST.
Access Point, Router, or Switch	Turris MOX router running OpenWRT	The Onboarding Router segments the onboarding device from the rest of the network until the BRSKI onboarding is complete

Component	Product	Function
Supply Chain Integration Service (Manufacturer Authorized Signing Authority—MASA)	Minerva Highway, which is a MASA provided by Sandelman Software Works	The device manufacturer provides device bootstrapping information (e.g., the device's X.509 certificate) and device ownership information to the MASA. The MASA creates and signs a voucher saying who the owner of the device is and provides this voucher to the IoT device via the Domain Registrar so that the device can verify that the network that is trying to onboard it is authorized to do so.
Authorization Service	Minerva Highway, which is a MASA provided by Sandelman Software Works	As described in the previous row.
IoT Device (Pledge)	Sandelman Software Works Reach Pledge Simulator	The device that is used to demonstrate trusted network-layer onboarding by joining the network.
Secure Storage	N/A (The IoT devices and the Sandelman Software Works Reach Pledge Simulator do not include secure storage)	Storage on the IoT device that is designed to be protected from unauthorized access and capable of detecting attempts to hack or modify its contents. Used to store and process private keys, credentials, and other information that must be kept confidential.
Certificate Authority	N/A (self-signed certificates were used)	Issues and signs certificates as needed.
Application-Layer Onboarding Service	None. Not supported in this build.	After connecting to the network, the device mutually authenticates with a trusted application service and interacts with it at the application layer.
Ongoing Device Authorization	N/A – Not intended for inclusion in this build	Performs activities designed to provide an ongoing assessment of the device's trustworthiness and authorization to access network resources. For example, it may perform behavioral analysis or device attestation and use the results to determine whether the device should be granted access to certain high-value resources, assigned to a particular network segment, or other action taken.
Manufacturer Factory Provisioning Process	N/A (Not implemented at the time of publication)	Manufactures the IoT device. Creates, signs, and installs the device's unique identity and other birth credentials into secure storage. Installs information the device requires for application-layer onboarding (if applicable). May populate a manufacturer database with information regarding devices that are created and, when the devices are sold, may record what entity owns them.

E.2 Build 3 Architecture

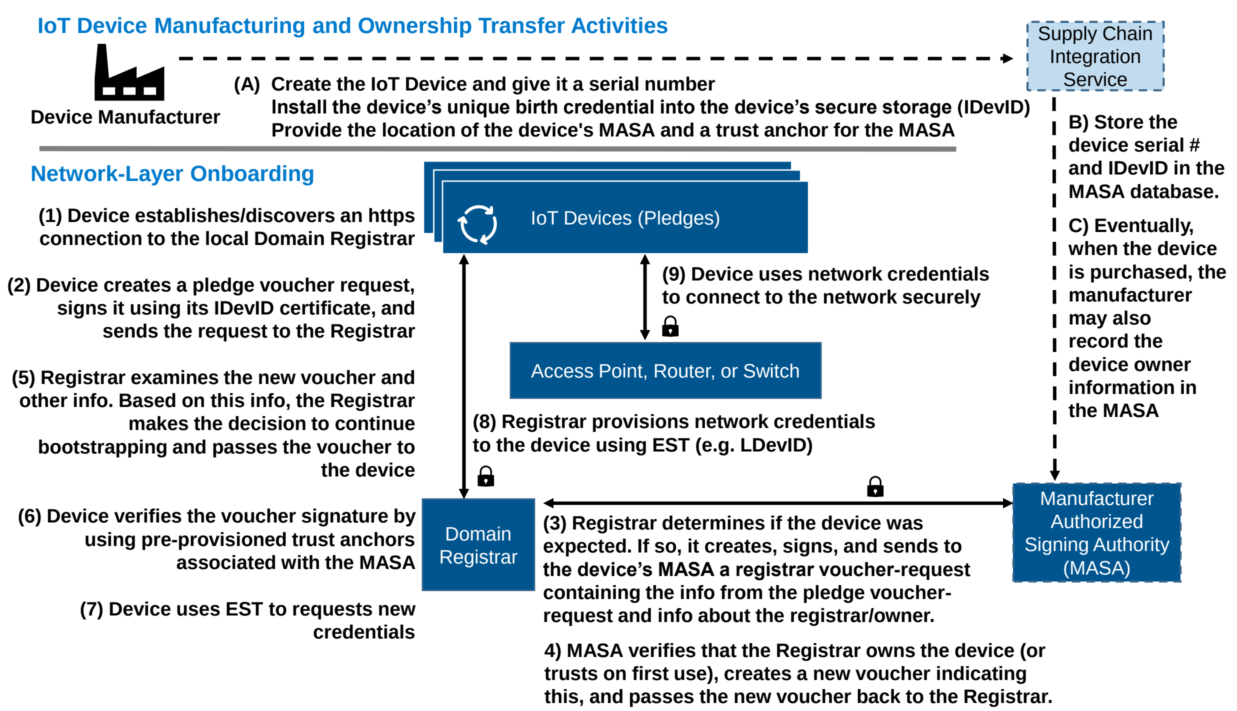
E.2.1 Build 3 Logical Architecture

The network-layer onboarding steps that are performed in Build 3 are depicted in Figure E-1. These steps are broken into two main parts: those required to transfer device bootstrapping information from the device manufacturer to the device owner's authorization service (labeled with letters) and those required to perform network-layer onboarding of the device (labeled with numbers). These steps are described in greater detail in IETF RFC 8995.

The device manufacturer:

1. Creates the device and installs a unique serial number and birth credential into secure storage on the device. This unique birth credential takes the form of a private key and its associated 802.1AR certificate, e.g., the device's IDevID. As part of this factory-installed certificate process, the location of the device's MASA is provided in an extension to the IDevID. The device is also provided with trust anchors for the MASA entity that will sign the returned vouchers.
2. Stores information about the device, such as its serial number and its IDevID, in the MASA's database.
3. Eventually, when the device is sold, the MASA may also record the device ownership information in its database.

Figure E-1 Logical Architecture of Build 3



After obtaining the device, the device owner provisions the device with its network credentials by performing the following network-layer onboarding steps:

1. The owner puts the device into onboarding mode. The device establishes an https connection to the local Domain Registrar. Trust in the Domain Registrar is provisional. (In a standard implementation, the device would use link-local network connectivity to locate a join proxy, and the join proxy would provide the device with https connectivity to the local Domain Registrar. The Build 3 implementation, however, does not support discovery at this time. To overcome this code limitation, the IoT device has been pre-provided with the address of the local Domain Registrar, to which it connects directly.)
2. The device creates a pledge voucher-request that includes the device serial number, signs this request with its IDevID certificate (i.e., its birth credential), and sends this signed request to the Registrar.
3. The Registrar receives the pledge voucher-request and considers whether the manufacturer is known to it and whether devices of that type are welcome. If so, the Registrar forms a registrar voucher-request that includes all the information from the pledge voucher-request along with information about the registrar/owner. The Registrar signs this registrar voucher-request. It locates the MASA that the IoT device is known to trust (e.g., the MASA that is identified in the device's IDevID extension) and sends the registrar voucher-request to the MASA.
4. The MASA consults the information that it has stored and applies policy to determine whether or not to approve the Registrar's claim that it owns and/or is authorized to onboard the device. (For example, the MASA may consult sales records for the device to verify device ownership, or it may be configured to trust that the first registrar that contacts it on behalf of a given device is in fact the device owner.) Assuming the MASA decides to approve the Registrar's claim to own and/or be authorized to onboard the device, the MASA creates a voucher that directs the device to accept its new owner/authorized network, signs this voucher, and sends it back to the Registrar.
5. The Registrar receives this voucher, examines it along with other related information (such as security posture, remote attestation results, and/or expected device serial numbers), and determines whether it trusts the voucher. Assuming it trusts the voucher, the Registrar passes the voucher to the device.
6. The device uses its factory-provisioned MASA trust anchors to verify the voucher signature, thereby ensuring that the voucher can be trusted. The voucher also validates the Registrar and represents the intended owner, ending the provisional aspect of the EST connection.
7. The device uses Enrollment over Secure Transport (EST) to request new credentials.
8. The Registrar provisions network credentials to the device using EST. These network credentials get stored into secure storage on the device, e.g., as an LDevID.
9. The device uses its newly provisioned network credentials to connect to the network securely.

This completes the trusted network-layer onboarding process for Build 3.

E.2.2 Build 3 Physical Architecture

Section 5.4 describes the physical architecture of Build 3.

Appendix F Build 4 (Thread, Silicon Labs-Thread, Kudelski KeySTREAM)

F.1 Technologies

Build 4 is an implementation of network-layer connection to an OpenThread network, followed by use of the Kudelski IoT keySTREAM Service to perform independent (see Section 3.3.2) application-layer onboarding of the device to a particular customer's tenancy in the AWS IoT Core. To join the network, the joining device generates and displays a pre-shared key that the owner enters on the commissioner, through a web interface, for authentication. The network-layer infrastructure for Build 4 was provided by Silicon Labs. The application-layer onboarding infrastructure for Build 4 was provided by Kudelski IoT. IoT devices that were onboarded using Build 4 were provided by Silicon Labs. For more information on these collaborators and the products and technologies that they contributed to this project overall, see Section 3.4.

Build 4 network infrastructure components within the NCCoE lab consist of a Thread border router (which is implemented using a Raspberry Pi) and a Silicon Labs Gecko Wireless Starter Kit. Build 4 also requires support from the Kudelski IoT keySTREAM service to perform application-layer onboarding. The keySTREAM service comes as a SaaS platform that is running in the cloud (accessible via the internet), and a software library (KTA – Kudelski Trusted Agent) that is integrated in the IoT device software stack. The KTA integrates with the Silicon Labs' Hardware Root of Trust (Secure Vault). The IoT device that is connected to the network and application-layer onboarded using Build 4 is the Silicon Labs Thunderboard (BRD2601A) with EFR32MG24x with Secure Vault(TM) High which is security certified to PSA/SESIP Level 3.

Table F-1 lists the technologies used in Build 4. It lists the products used to instantiate each logical build component and the security function that the component provides. The components are logical. They may be combined in physical form, e.g., a single piece of hardware may house a network onboarding component, a router, and a wireless access point.

Table F-1 Build 4 Products and Technologies

Component	Product	Function
Network-Layer Onboarding Component (Thread Protocol Component)	SLWSTK6023A Thread Radio Transceiver (Wireless starter kit);	The SLWSTK6023A acts as a Thread radio transceiver or radio coprocessor (RCP), allowing the open thread boarder router host platform to form and communicate with a Thread network. If the Thread MeshCoP were running on this device, it would provision the IoT device with credentials for the Thread network.
Access Point, Router, or Switch	OpenThread Border Router (OTBR) hosted on a Raspberry Pi	Router that has interfaces both on the Thread network and on the IP network to act as a bridge between the Thread network and the public internet. This allows the IoT device that communicates using the Thread wireless protocol to communicate with cloud services.

Component	Product	Function
Supply Chain Integration Service	Silicon Labs Custom Parts Manufacturer Service (CPMS)	To support network-layer onboarding, the device manufacturer provides device bootstrapping information to the to the device owner.
Authorization Service	Not implemented	Enables the network to verify that the device that is trying to onboard to it is authorized to do so.
IoT Device	Silicon Labs Thunderboard (BRD2601A)	The IoT device that is used to demonstrate trusted network- and application-layer onboarding.
Secure Storage	Secure Vault ™ High on Silicon Labs IoT device	Storage designed to be protected from unauthorized access and capable of detecting attempts to hack or modify its contents. Used to store and process private keys and other information that must be kept confidential.
Certificate Authority	Each tenant in the Kudelski keySTREAM service cloud has its own certificate signing authority	Issues and signs certificates as needed. For application-layer onboarding, the device owner has its own certificate signing authority in its portion of the Kudelski keySTREAM service cloud.
Application-Layer Onboarding Service	Kudelski keySTREAM Service	After connecting to the Thread network, the device performs application-layer onboarding by accessing the Kudelski keySTREAM service. The device and the keySTREAM service mutually authenticate; the keySTREAM service verifies the device's owner, generates an application-layer credential (i.e., an AWS certificate that is based on the device's chipset identity and owner) for the device, and provisions the device with this X.509 credential that will enable the device to access the owner's tenancy in the AWS IoT Core cloud.
Ongoing Device Authorization	N/A – Not intended for inclusion in this build	Performs activities designed to provide an ongoing assessment of the device's trustworthiness and authorization to access network resources. For example, it may perform behavioral analysis or device attestation and use the results to determine whether the device should be granted access to certain high-value resources, assign the device to a particular network segment, or take other action.

Component	Product	Function
Manufacturer Factory Provisioning Process	Silicon Labs Custom Parts Manufacturing Service (CPMS)	Manufactures the IoT device. Creates, signs, and installs the device's unique identity and other birth credentials into secure storage. Installs software and information the device requires for application-layer onboarding. May populate a manufacturer database with information regarding devices that are created and, when the devices are sold, may record what entity owns them. The MG24 "B" version comes pre-loaded with a Silicon Labs Birth certificate. The "A" or "B" version birth certificate can be modified via their Custom Part Manufacturing Service (CPMS) to be unique per end device manufacturer and signed into their Root CA if desired.

F.2 Build 4 Architecture

F.2.1 Build 4 Logical Architecture

Build 4 demonstrates a device connecting to an OpenThread network. IoT devices generate and use a pre-shared key to connect to the OpenThread network of Build 4 using the Thread MeshCoP service. Once a device is connected to the OpenThread network of Build 4, it gets access to an IP network via a border router, and then performs application-layer onboarding using the Kudelski keySTREAM Service. Kudelski keySTREAM is a device security management service that runs as a SaaS platform on the Amazon cloud. Build 4 relies on an integration that has been performed between Silicon Labs and Kudelski keySTREAM. KeySTREAM has integrated software libraries with the Silicon Lab EFR32MG24 (MG24) IoT device's secure vault to enable the private signing key that is associated with an application-layer certificate to be stored into the secure vault using security controls that are available on the MG24. This integration ensures that application-layer credentials can be provisioned into the vault securely such that no key material is misused or exposed.

At a high level, the steps required to enable demonstration of Build 4's network connection and application-layer onboarding capabilities can be broken into the following three main parts:

- Device Preparation: The IoT device is prepared for network connection and application-layer onboarding by the device manufacturer.
    - The device comes from the manufacturer ready to be provisioned onto a Thread network. No additional preparation is required.
    - The device is prepared for application-layer onboarding on behalf of a specific, pre-defined customer who will become its owner. The device is assigned ownership to this customer (e.g., customer A) and this ownership information is sealed into device firmware, permanently identifying the device as being owned by customer A. The device owner, customer A, has a tenancy on the Kudelski keySTREAM Service and is also an Amazon Web Services (AWS) customer. After the device has been prepared, the device is provided to its owner (customer A).

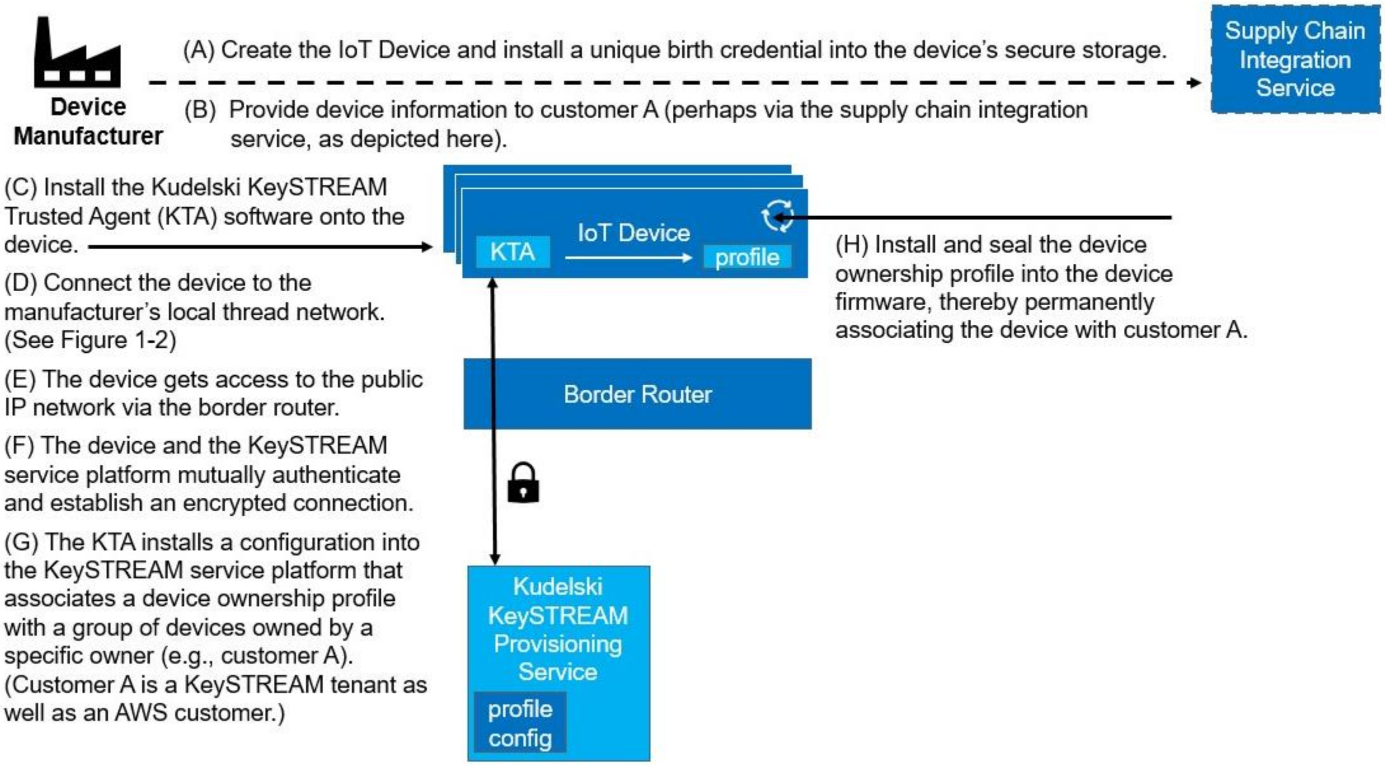
- Network Connection: Customer A connects the device to Customer A’s OpenThread network by entering the pre-shared key displayed on the device’s serial terminal in the OpenThread Border Router’s (OTBR) web interface. This allows the network‘s radio channel, PAN ID, extended PAN ID and network name to be discovered, avoiding the need to preconfigure any of these parameters. Once on customer A’s OpenThread network, the device has access to the public IP network via the border router.
- Application-Layer Onboarding: The device and the keySTREAM service mutually authenticate, keySTREAM confirms that customer A owns the device, and keySTREAM provisions the device with an AWS certificate that is specific to the device and to customer A, enabling the device to authenticate to customer A’s tenancy in the AWS IoT Core.

Each of these three aspects of the demonstration are illustrated in its own figure and described in more detail in the three subsections below.

F.2.1.1 Device Preparation

Figure F-1 depicts the steps that are performed by the device manufacturer, which in this case is Silicon Labs, to prepare the device for network- and application-layer onboarding by a particular customer, Customer A. Each step is described in more detail below. Because these steps are performed to prepare the device for onboarding rather than as part of onboarding itself, they are labeled with letters instead of numbers in keeping with the conventions used in other build descriptions.

Figure F-1 Logical Architecture of Build 4: Device Preparation



The following steps are performed to prepare the device for network connection and application-layer onboarding:

1. The manufacturer creates the device, which in this case is a Silicon Labs MG24, and prepares it for network connection by installing the device's unique birth credential into the device's chipset. This chipset identity is a hardware root of trust. The MG24 "B" version comes pre-loaded with a Silicon Labs Birth certificate. The "A" or "B" version birth certificate can be modified via their Custom Part Manufacturing Service (CPMS) to be unique per end device manufacturer and signed into their Root CA if desired.

2. The manufacturer provides information about the device to customer A (perhaps via the supply chain service, as depicted in Figure 1-1) so customer A can be aware that the device is expected on its network.

3. The manufacturer prepares the device for application-layer onboarding by installing the Kudelski keySTREAM Trusted Agent (KTA) software onto the device.

4. The manufacturer connects the device to the manufacturer's local OpenThread network. (See Figure 1-2 for details of the network connection steps.) Note that in this case, which is the first time that the device is being connected to a network, the device is being connected to the manufacturer's network rather than to the network of the device's eventual owner.

5. After the device connects to the manufacturer's OpenThread network, the device has access to the public IP network via the border router.

6. The device and the Kudelski keySTREAM service mutually authenticate and establish an encrypted connection.

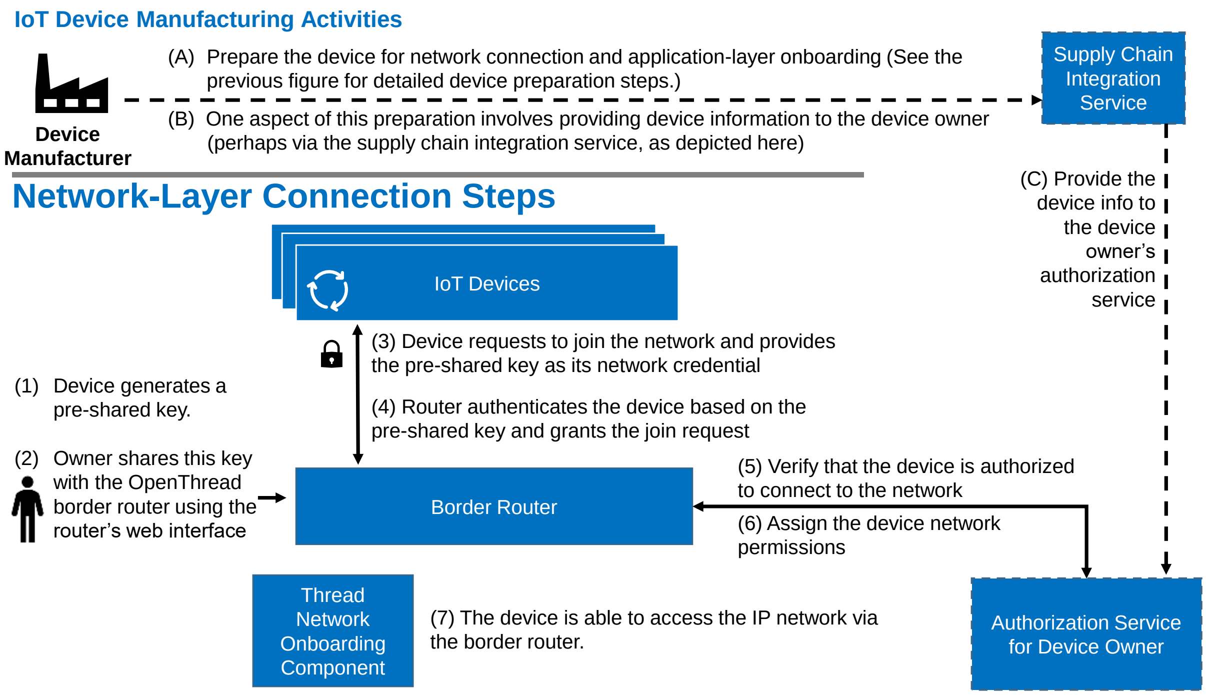
7. The KTA installs a configuration into the keySTREAM service platform that builds up a group of devices that belong to a certain end user and associates the group with a device ownership profile. This device ownership profile is associated with a particular customer (e.g., customer A). The same device profile is used by all devices in a group of devices that are owned by this owner. The profile is not specific to individual devices. The owner of these devices (customer A) has a keySTREAM tenancy, which includes a dedicated certificate signing CA. Customer A is also an AWS customer.

8. The device manufacturer installs and seals this device ownership profile into the device firmware. This profile permanently identifies the device as being owned by customer A.

F.2.1.2 Network-Layer Connection

Figure F-2 depicts the steps of an IoT device connecting to that thread network using a pre-shared key that the device generates and shares with the OpenThread boarder router. Each step is described in more detail below.

Figure F-2 Logical Architecture of Build 4: Connection to the OpenThread Network



The device connects to the OpenThread network using the following steps:

1. The device generates a pre-shared key.

2. The owner starts the commissioning process by entering this pre-shared key on the OpenThread border router.

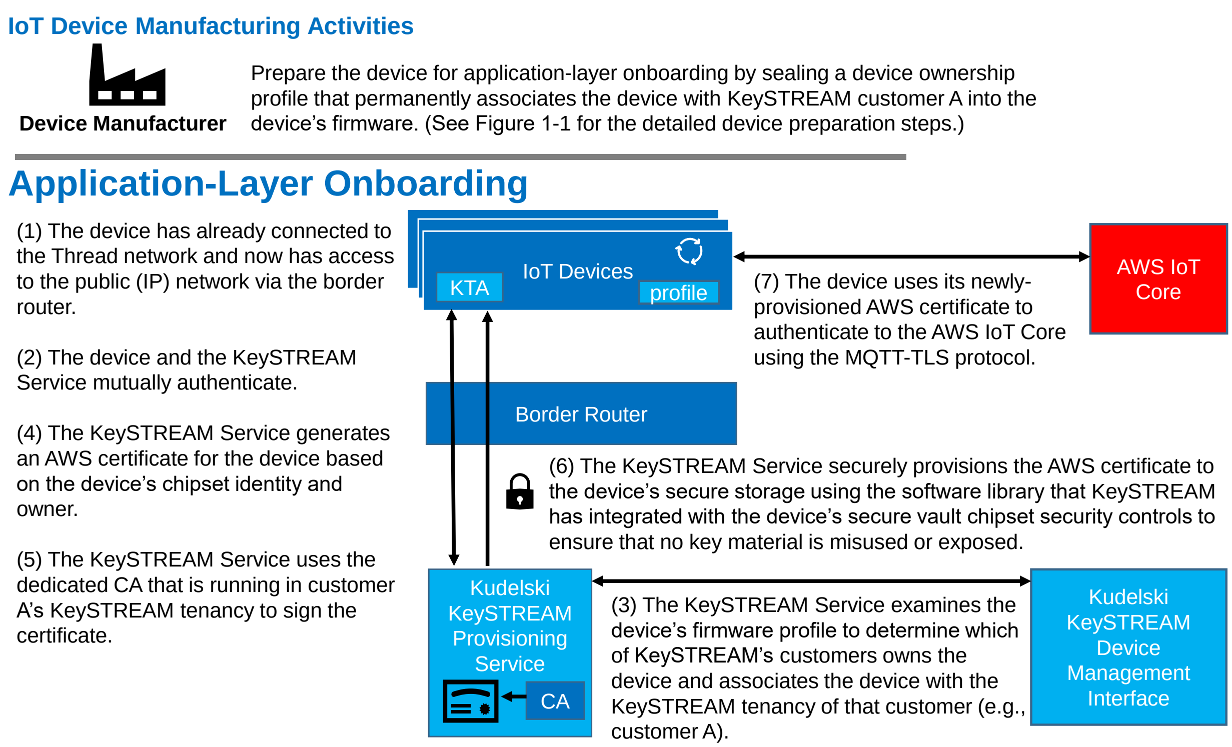
3. The device requests to join the network and provides the pre-shared key as its network credential.
4. The network authenticates the device based on the pre-shared key and grants the join request.
5. The network verifies that the device is authorized to connect to the network.
6. The network assigns the device network permissions and configures these as policies on the border router.
7. The device is able to access the IP network (and the internet) via the border router.

This completes the network-layer connection process.

F.2.1.3 Application-Layer Onboarding

Figure F-3 depicts the steps of the application-layer onboarding process using the Kudelski keySTREAM service. Each step is described in more detail below.

Figure F-3 Logical Architecture of Build 4: Application-Layer Onboarding using the Kudelski keySTREAM Service



The application-layer onboarding steps performed to provision the device with its application-layer credentials (e.g., its AWS certificate) are as follows:

1. The device, which is already connected to the OpenThread network, accesses the IP network via the border router.
2. The device and the keySTREAM service mutually authenticate.

3. The keySTREAM Service examines the device's firmware profile to determine which of keySTREAM's customers owns the device. In this case, customer A is identified as the device owner. The keySTREAM service associates the device with customer A's keySTREAM tenancy.
4. The keySTREAM Service generates an AWS IoT Core certificate for the device based on both the device's ownership information and the secure hardware root of trust that is in the device's chipset.
5. The keySTREAM Service uses the dedicated CA that is running in customer A's keySTREAM tenancy to sign the AWS certificate.
6. The keySTREAM Service securely provisions the AWS certificate to the device's secure storage using the software library that keySTREAM has integrated with the device's secure vault chipset security controls to ensure that no key material is misused or exposed.
7. The device uses its newly provisioned application-layer credentials (i.e., the AWS certificate) to authenticate to customer A's tenancy in the AWS IoT Core using the MQTT-TLS protocol.

F.2.2 Build 4 Physical Architecture

Section 5.5 describes the physical architecture of Build 4.

Appendix G Build 5 (BRSKI over Wi-Fi, NquiringMinds)

G.1 Technologies

Build 5 is an implementation of network-layer onboarding that uses a version of the BRSKI Protocol that has been modified to work over Wi-Fi. After the IoT device has joined the network, Build 5 also demonstrates a number of mechanisms that are performed on an ongoing basis to provide continuous, policy-based authorization and assurance. Both the network-layer onboarding infrastructure and the continuous assurance service for Build 5 were provided by NquiringMinds. This entire build can be replicated using the open sourced TrustNetZ code base.

For more information on NquiringMinds and the products and technologies that they contributed to this project overall, see Section 3.4.

Build 5 network onboarding infrastructure components within the NCCoE lab consist of a Linux based Raspberry Pi 4B router (which also runs the registrar service and MASA service), and a USB hub. The Build 5 components used to support the continuous assurance service include TrustNetZ Authorization interfaces, TrustNetZ information provider, and TrustNetZ policy engine. The IoT devices that are onboarded using Build 5 are a Raspberry Pi device. These IoT devices do not have secure storage, but use the Infineon Optiga SLB 9670 TPM 2.0 as an external secure element. Build 5 depends on an IDevID (X.509 Certificate) having been provisioned to the secure element of the IoT device (pledge) prior to onboarding, as part of the factory provisioning process (see Section H.1). For Build 5, this factory provisioning process was accomplished by the BRSKI Factory Provisioning Build, which is described in Appendix H.3.

Table G-1 lists the technologies used in Build 5. It lists the products used to instantiate each logical build component and the security function that the component provides. The components are logical. They may be combined in physical form, e.g., a single piece of hardware may house a network onboarding component, a router, and a wireless access point.

Table G-1 Build 5 Products and Technologies

Component	Product	Function
Network-Layer Onboarding Component (BRSKI Domain Registrar)	Stateful, non-persistent Linux app that has two functional interfaces for both BRSKI and for the Authentication Service. (TrustNetZ onboarding)	Runs the BRSKI protocol modified to work over Wi-Fi and acts as a BRSKI Domain Registrar. It authenticates the IoT device, receives a voucher request from the IoT device, and passes the request to the MASA. It also receives a voucher from the MASA, verifies it, and passes it to the IoT device. Assuming the IoT device finds the voucher to be valid and determines that the network is authorized to onboard it, the Domain Registrar provisions network credentials to the IoT device using EST.

Component	Product	Function
Access Point, Router, or Switch	Raspberry Pi 4B equipped with USB Wi-Fi dongle, running TrustNetZ AP code.	Router, providing an open Wi-Fi network and closed Wi-Fi network. Physical access control is mediated through the RADUIS interface (which is part of the TrustNetZ AP configuration) The AP also receives network commands from the continuous assurance service.
Supply Chain Integration Service (Manufacturer Authorized Signing Authority—MASA)	TrustNetZ MASA	The MASA creates and signs a voucher and provides this voucher to the IoT device via the Registrar so that the device can verify that the network that is trying to onboard it is authorized to do so.
Authorization Service	Linux application which contains an encapsulated policy engine (TrustNetZ policy engine)	Determines whether the device is authorized to be onboarded to the network. The application features a REST API which accepts verifiable credential claims to feed data on entities and their relationships into its SQL database. The policy engine itself is based on verifiable credentials presentation, (persisted to SQL database), making it easily configurable and extensible.
IoT Device	Raspberry Pi devices (running TrustNetZ pledge agent)	The IoT device that is used to demonstrate trusted network- and application-layer onboarding. Handles the client side BRSKI protocols, the integration with the secure storage, with factory provisioning and TLS connections.
Secure Storage	Infineon Optiga SLB 9670 TPM 2.0	Storage on the IoT device that is designed to be protected from unauthorized access and capable of detecting attempts to hack or modify its contents. Used to store and process private keys and other information that must be kept confidential.
Certificate Authority	TrustNetZ demo manufacturer CA (MPR – manufacture provisioning root) TrustNetZ Domain CA	Two CA are used in Build 5 Domain CA issues certificates and provides signing and attestation functions that model network owner relationships (e.g. sign the LDevID certificate) Manufacturer CA issues the IDevID certificates; proving the device has been created by the manufacturer.
Application-Layer Onboarding Service	TrustNetZ Demo application sever	After connecting to the network, the device mutually authenticates with a trusted application service and interacts with it at the application layer. The IDevID and TPM private key are used to establish a TLS session with the demonstration application server and send data to it from the device. This demonstrates the concept of secure connection to a third-party application server using the cryptographic artifacts from the onboarding process.

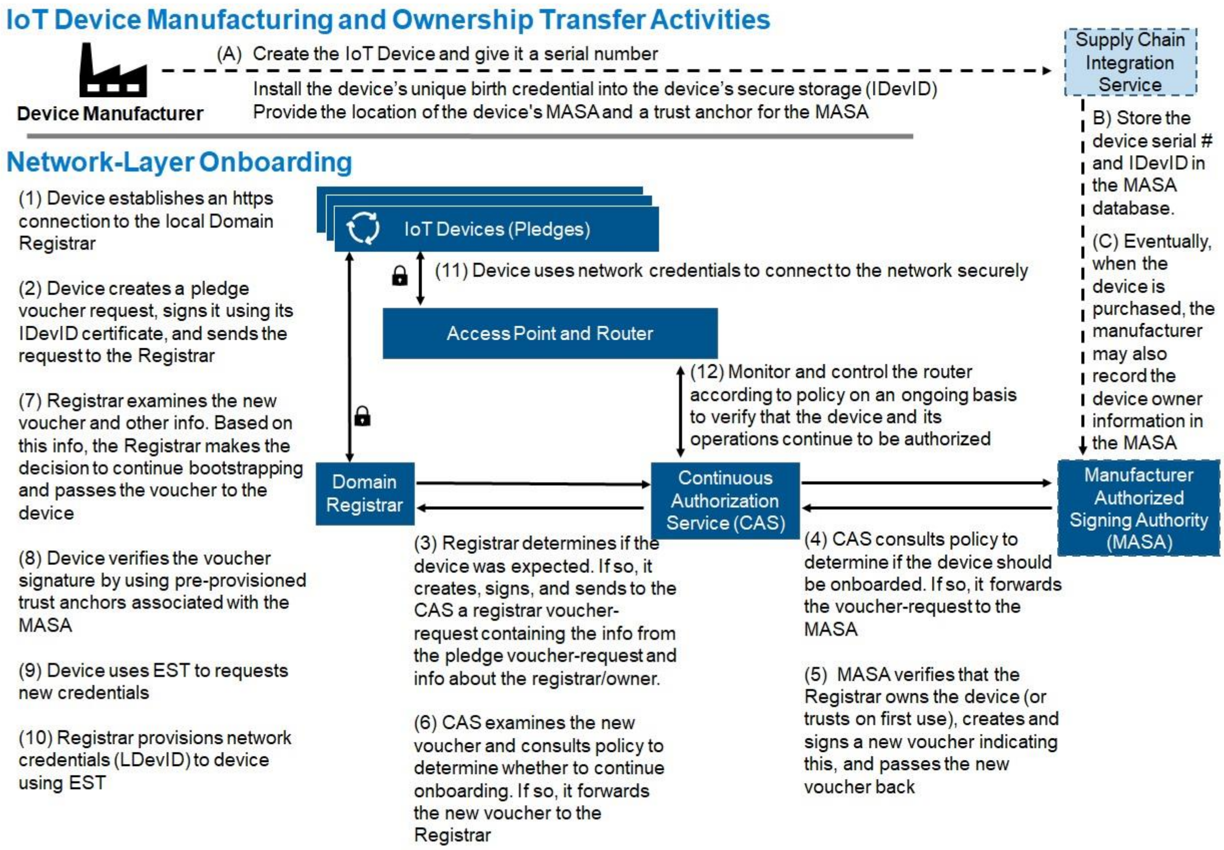
Component	Product	Function
Ongoing Device Authorization	Continuous Authorization Service, which calls into the in the TrustNetZ policy engine	Designed to perform a set of ongoing, policy-based continuous assurance and authorization checks on the device after it has connected to the network. As of this publication, the following ongoing checks have been implemented: ▪ The manufacturer of the device must be trusted by the network owner ▪ The device must be trusted by a user with appropriate privileges ▪ The device must have an associated device type ▪ The vulnerability score of the software bill of materials (SBOM) for the device type must be lower than a set threshold ▪ The device must not have contacted an IP address that is on a deny list If it fails any of these periodic checks, its voucher is revoked, which removes the device from the network.
Manufacturer Factory Provisioning Process	BRSKI Factory Provisioning Process used to provision the Infineon TPM with its private key and IDevID (See Appendix H.3)	Manufactures the IoT device. Creates, signs, and installs the device's unique identity (i.e., its IDevID, which is an X.509 certificate) into secure storage. Installs information the device requires for application-layer onboarding. Populates the MASA with information regarding devices that are created and, when the devices are sold, may record what entity owns them.

G.2 Build 5 Architecture

G.2.1 Build 5 Logical Architecture

The network-layer onboarding steps that are performed in Build 5 are depicted in Figure G-1. These steps are broken into two main parts: those required to transfer device bootstrapping information from the device manufacturer to the MASA (labeled with letters) and those required to perform network-layer onboarding of the device and establish the operation of the continuous authorization service (labeled with numbers).

Figure G-1 Logical Architecture of Build 5



The device manufacturer:

1. Creates the device and installs a unique serial number and birth credential into secure storage on the device. This unique birth credential takes the form of a private key and its associated 802.1AR certificate, e.g., the device's IDevID. As part of this factory-installed certificate process, the location of the device's manufacturer authorized signing authority (MASA) is provided in an extension to the IDevID. The device is also provided with trust anchors for the MASA entity that will sign the returned vouchers.

2. Stores information about the device, such as its serial number and its IDevID, in the MASA's database.

3. Eventually, when the device is sold, the MASA may also record the device ownership information in its database.

After obtaining the device, the device owner provisions the device with its network credentials by performing the following network-layer onboarding steps:

1. The owner puts the device (i.e., the pledge) into onboarding mode. The device establishes an https connection to the local Domain Registrar. (In a standard BRSKI implementation, the device would have wired network connectivity. The device would use its link-local network connectivity to locate a join proxy, and the join proxy would provide the device with https connectivity to the

local Domain Registrar.) The Build 5 implementation, however, relies on wireless connectivity and initially uses the unauthenticated EAP-TLS protocol. The pledge discovers potential onboarding networks by searching for public Wi-Fi networks that either match a particular SSID wildcard name or that advertise a particular realm. When the device finds a potential onboarding network, it connects to it and attempts to discover the registrar. The pledge will connect to the open Wi-Fi network and will receive either an IPv4 or IPv6 address. Subsequently, the pledge will listen to mDNS packets and will obtain the list of join proxies (IP addresses). Finally, the pledge will subsequently connect to each join proxy using the BRSKI-EST protocol.

2. The device creates a pledge voucher-request that includes the device serial number, signs this request with its IDevID certificate (i.e., its birth credential), and sends this signed request to the Registrar.

3. The Registrar receives the pledge voucher-request and considers whether the manufacturer is known to it and whether devices of that type are welcome. If so, the Registrar forms a registrar voucher-request that includes all the information from the pledge voucher request along with information about the registrar/owner. The Registrar sends this registrar voucher-request to the Continuous Authorization Service.

4. The Continuous Authorization Service consults policy to determine if this device should be permitted to be onboarded and what other conditions should be enforced. An example of policy that might be used is that the network owner wants to disable MASA validation. Assuming the device is permitted to be onboarded, the Continuous Authorization Service locates the MASA that the IoT device is known to trust (i.e., the MASA that is identified in the device's IDevID extension) and sends the registrar voucher-request to the MASA.

5. The MASA consults the information that it has stored and applies policy to determine whether to approve the Registrar's claim that it owns the device. (For example, the MASA may consult sales records for the device to verify device ownership, or it may be configured to trust that the first registrar that contacts it on behalf of a given device is in fact the device owner). Assuming the MASA decides to approve the Registrar's claim to own the device, the MASA creates a new voucher that directs the device to accept its new owner, signs this voucher, and sends it back to the Continuous Authorization Service.

6. The Continuous Authorization Service receives this new voucher and examines it in consultation with policy to determine whether to continue onboarding. Some examples of policies that might be used include: permit onboarding only if no current critical vulnerabilities have been disclosed against the declared device type, the device instance has successfully passed a site-specific test process, or a test compliance certificate has been found for the declared device type. Assuming the device is permitted to be onboarded, the Continuous Authorization Service sends the new voucher to the Domain Registrar.

7. The Domain Registrar receives and examines the new voucher along with other related information and determines whether it trusts the voucher. Assuming it trusts the voucher, the Registrar passes the voucher to the device.

8. The device uses its factory-provisioned MASA trust anchors to verify the voucher signature, thereby ensuring that the voucher can be trusted.
9. The device uses Enrollment over Secure Transport (EST) to request new credentials.
10. The Registrar provisions network credentials to the device using EST. These network credentials get stored into secure storage on the device, e.g., as an LDevID.
11. The device uses its newly provisioned network credentials to connect to the network securely.
12. After the device is connected and begins operating on the network, the Continuous Authorization Service and the router make periodic asynchronous calls to each other that enable the Continuous Authorization Service to monitor device behavior and constrain communications to and from the device as needed in accordance with policy. In this manner, the Continuous Authorization Service interacts with the router on an ongoing basis to verify that the device and its operations continue to be authorized throughout the device's tenure on the network.

This completes the network-layer onboarding process for Build 5 as well as the initialization of the Build 5 continuous authorization service. More details regarding the Build 5 implementation can be found at https://trustnetz.nqm.ai/docs/.

G.2.2 Build 5 Physical Architecture

Section 5.6 describes the physical architecture of Build 5.

Appendix H Factory Provisioning Process

H.1 Factory Provisioning Process

The Factory Provisioning Process creates and provisions a private key into the device's secure storage; generates and signs the device's certificate (when BRSKI is supported), generates the device's DPP URI (when Wi-Fi Easy Connect is supported), or generates other bootstrapping information (when other trusted network-layer onboarding protocols are supported); provisions the device's certificate, DPP URI, or other bootstrapping information onto the device; and sends the device's certificate, DPP URI, or other bootstrapping information to the manufacturer's database, which will eventually make this information available to the device owner to use during network-layer onboarding.

H.1.1 Device Birth Credential Provisioning Methods

There are various methods by which a device can be provisioned with its private key and bootstrapping information (e.g., its certificate, DPP URI, etc.) depending on how, where, and by what entity the public/private key pairs are generated [14]. Additional methods are also possible depending on how the device's certificate is provided to the manufacturer's database. The following are high-level descriptions of five potential methods for provisioning device birth credentials during various points in the device lifecycle. These methods are not intended to be exhaustive:

1. **Method 1: Key Pair Generated on IoT Device**
   Summary: Generate the private key on the device; device sends the device's bootstrapping information (e.g., the device's certificate or DPP URI) to the manufacturer's database. The steps for Method 1 are:
   a. The public/private key pair is generated on the device and stored in secure storage.
   b. The device generates and signs a CSR structure and sends the CSR to the manufacturer's IDevID CA, which sends a signed certificate (IDevID) back to the device.
   c. If BRSKI is being supported, the device loads the certificate (IDevID) into its secure storage; if Wi-Fi Easy Connect is being supported, the device creates a DPP URI and loads that into secure storage.
   d. The device sends the certificate or DPP URI to the manufacturer's database.

   One disadvantage of this method is that the device's random number generator is being relied upon to generate the key pair, and it is possible that a device's random number generator will not be as robust as the random number generator that would be included in an SE, for example. An advantage of this method is that the device's private key is not vulnerable to disclosure, assuming the device is equipped with a strong random number generator that is used for key generation and the private key is put into secure storage immediately upon generation.

2. **Method 2: Key Pair Generated in Secure Element**
   Summary: Generate the private key in a secure element on the device; IDevID CA provides the device certificate to the manufacturer's database. The steps for Method 2 are:
   a. The public/private key pair is generated within the device's SE.

b. The device generates a CSR structure, the SE signs it, and the device sends the CSR to the manufacturer's IDevID CA, which sends a signed certificate (IDevID) back to the device.
c. If BRSKI is being supported, the device loads the certificate (IDevID) into its secure storage; if Wi-Fi Easy Connect is being supported, the device creates a DPP URI and loads that into secure storage.
d. The IDevID CA provides the certificate to the manufacturer's database. The manufacturer stores either the certificate (i.e., if BRSKI is being supported), or creates and stores a DPP URI (i.e., if Wi-Fi Easy Connect is being supported).

Method 2 is similar to Method 1 except that in method 2, the key pair is generated and stored in a secure element and the manufacturer's database receives the signed certificate directly from the CA (either via a push or a pull) rather than via the device. An advantage of method 2 is that the device's private key is not vulnerable to disclosure because secure elements are normally equipped with a strong random number generator and tamper-proof storage.

3. Method 3: Key Pair Loaded into IoT Device

Summary: Generate the private key in the device factory and load it onto the device. The steps for Method 3 are:

a. The public/private key pairs and certificates are generated in advance at the device factory and recorded in the manufacturer's database.
b. The public/private key pair and certificate are loaded onto the device at the device factory.

One advantage of this method is that there is no need to trust the random number generator on the device to generate strong public/private key pairs. However, the private keys may be vulnerable to disclosure during the period of time before they are provisioned into secure storage on the devices (and afterwards if they are not deleted once they have been copied into secure storage).

4. Method 4: Key Pair Pre-Provisioned onto Secure Element

Summary: Generate the private key in the SE and load the certificate on the device at the SE factory (SEF). The steps for Method 4 are:

a. The public/private key pair and certificate are generated in advance in the SE at the SEF and the public key is recorded.
b. The certificate is loaded onto the devices at the SEF.
c. The certificates and the serial numbers of their corresponding devices are provided to the device manufacturer, and the device manufacturer can put them into the manufacturer database.
d. The CA that signs the certificates that are generated and loaded onto the SEs may come from either the SEF or the device manufacturer. (Note: the CA is likely not located at the factory, which may be offshore.)

Additional trust anchors can also be loaded into the SE at the SEF (e.g., code signing keys, server public keys for TLS connections, etc.) As with methods 2 and 3, one advantage of this method (method 4) is that there is no need to trust the random number generator on the device to generate strong public/private key pairs because the random number generator on the SE is used instead. With this method, the security level of the manufacturer's factory does not need to be as high as that of the SEF because all key generation and certificate signing is performed at the SEF; the manufacturer can rely on the security of the SEF, which can be advantageous to the device manufacturer, assuming that the SEF is in fact secure.

5. **Method 5: Private Key Derived from Shared Seed**

Summary: The device's private key is derived from a shared seed. The steps for Method 5 are:

a. The chip vendor embeds a random number into each IoT device (e.g., this may be burned into fuses on the IoT device, inside the Trusted Execution Environment (TEE)).
b. The IoT device manufacturer gets a copy of this seed securely (e.g., on a USB device that is transported via trusted courier).
c. On first boot, the IoT device generates a private key from this seed.
d. The manufacturer uses the same seed to generate a public key and signs a certificate.

As with method 4, with this option (method 5), there is no need for the IoT device manufacturer to have a secure factory because the IoT device manufacturer may rely on the security of the chip manufacturer. However, the IoT device manufacturer must also rely on the security of the courier or other mechanism that is delivering the seed, and the IoT device manufacturer must ensure that the value of this seed is not disclosed.

H.2 Factory Provisioning Builds – General Provisioning Process

The Factory Provisioning Builds implemented as part of this project simulate activities performed during the IoT device manufacturing process to securely provision the device's birth credentials (i.e., its private key) into secure storage on the device and make the device's network-layer bootstrapping information available by enrolling the device's public key into a database that will make this public key accessible to the device owner in a form such as a certificate or DPP URI. The method used in the factory provisioning builds most closely resembles *Method 2: Key Pair Generated on IoT Devic*e, as described in Section H.1.1.

There are several different potential versions of the factory provisioning build architecture depending on whether the credentials being generated are designed to support BRSKI, Wi-Fi Easy Connect, Thread, or some other trusted network-layer onboarding protocol. For example, when BRSKI is being supported, the device bootstrapping information that is created takes the form of an 802.1AR certificate (IDevID); if DPP is supported, it takes the form of a DPP URI.

Because this project does not have access to a real factory or the tools necessary to provision birth credentials directly into device firmware, the factory builds simulate the firmware loading process by loading factory provisioning code into the IoT device (e.g., a Raspberry Pi device). This code plays the role of the factory in the builds by instructing the SE that is attached to the IoT device to generate the device's private key and bootstrapping information. Once the IoT device has been provisioned with its birth credentials in this manner, it can, in theory, be network-layer onboarded to one of the project build networks.

H.3 BRSKI Factory Provisioning Builds (NquiringMinds and SEALSQ)

Two variants of the BRSKI Factory Provisioning Build were implemented:

- **NquiringMinds and SEALSQ implementation** (first version): SEALSQ, a subsidiary of WISeKey, and NquiringMinds collaborated to implement one version of the BRSKI Factory Provisioning Build. This build is designed to provision birth credentials to a Raspberry Pi device that has an attached secure element provided by SEALSQ.
- **NquiringMinds and Infineon implementation** (second version): NquiringMinds implemented a second version of the BRSKI Factory Provisioning Build using an Infineon SE. This build is designed to provision birth credentials to a Raspberry Pi device that has an attached Infineon Optiga SLB 9670 TPM 2.0.

H.3.1 BRSKI Factory Provisioning Build Technologies

The general infrastructure for the first version of the BRSKI Factory Provisioning Build (i.e., the NquiringMinds and SEALSQ implementation) is provided by SEALSQ. The first version of the BRSKI Factory Provisioning Build infrastructure consists of:

- A SEALSQ VaultIC SE that is attached to the Raspberry Pi
- SEALSQ Factory Provisioning Code that is located on an SD card and that communicates with the chip in the SE to
  - create a P-256 Elliptic Curve public/private key pair within the SE,
  - construct a certificate signing request, and
  - store the certificate in the SE as well as send it to the manufacturer’s database
- SEALSQ INeS CMS CA, a certificate authority for signing the device’s birth certificate

As mentioned earlier, separate factory provisioning builds are required for each network-layer onboarding protocol being supported. A small amount of factory provisioning code is required to be customized for each build, depending on the onboarding protocol that is supported and how the bootstrapping information will be provided to the manufacturer. In this build, NquiringMinds provided this code and made it available to the Raspberry Pi IoT device by placing it on an SD card. (This could be either in a partition of the SD card that holds the device’s BRSKI onboarding software or on a separate SD card altogether).

Table H-1 lists the technologies used in the first version of the BRSKI Factory Provisioning Build. It lists the products used to instantiate each logical build component and the security function that the component provides. The components listed are logical. They may be combined in physical form, e.g., a single piece of hardware may both generate key pairs and provide secure storage.

Table H-1 First Version of the BRSKI Factory Provisioning Build Products and Technologies

Component	Product	Function
Key Pair Generation Component	SEALSQ VaultIC and associated provisioning code	Generates and installs the public/private key pair into secure storage. The VaultIC has a SP800-90B certified random number generator for key pair generation.

Component	Product	Function
		[15][16][17] Signs the certificate signing request that is sent to the CA.
Secure Storage	SEALSQ VaultIC	Storage on the IoT device that is designed to be protected from unauthorized access and capable of detecting attempts to hack or modify its contents. Used to generate, store, and process private keys, credentials, and other information that must be kept confidential.
General Factory Provisioning Instructions	SEALSQ Factory Provisioning Code	Creates a CSR associated with the key pair, installs the signed certificate into secure storage. Creates a record of devices that it has created and their certificates.
Build-specific Factory Provisioning Instructions	NquiringMinds Factory Provisioning Code	Sends device ownership information and the certificate received by the General Factory Provisioning code to the MASA.
Manufacturer Database	MASA	When devices are manufactured, device identity and bootstrapping information is stored here by the manufacturer. Eventually, this database makes the device's bootstrapping information available to the device owner. Device bootstrapping information is information that the device owner requires to perform trusted network-layer onboarding; for BRSKI, the bootstrapping information is a signed certificate that is sent to the MASA, along with information regarding the device's owner.
Certificate Authority (CA)	SEALSQ INeS CMS CA	Issues and signs certificates as needed.

The second version of the BRSKI Factory Provisioning Build (i.e., the NquiringMinds implementation with an Infineon SE) infrastructure consists of:

- An Infineon Optiga SLB 9670 TPM 2.0. that is attached to the Raspberry Pi
- Factory Provisioning Code written by NquiringMinds that is located on an SD card and that communicates with the chip in SE to
    - create a P-256 Elliptic Curve public/private key pair within the SE,
    - construct a certificate signing request, and
    - store the certificate in the SE as well as send it to the manufacturer's database
- NquiringMinds Manufacturer Provisioning Root (MPR) server, which signs the device's IDevID birth certificate. It sits in the cloud and is securely contacted using the keys in the Infineon Optiga secure element.

In this build, NquiringMinds provided all of the factory provisioning code and made it available to the Raspberry Pi IoT device by placing it on an SD card. (This could be either in a partition of the SD card that holds the device's BRSKI onboarding software or on a separate SD card altogether).

Table H-2 lists the technologies used in the second version of the BRSKI Factory Provisioning Build. It lists the products used to instantiate each logical build component and the security function that the component provides. The components listed are logical. They may be combined in physical form, e.g., a single piece of hardware may both generate key pairs and provide secure storage.

Table H-2 Second Version of the BRSKI Factory Provisioning Build Products and Technologies

Component	Product	Function
Key Pair Generation Component	Infineon TPM and associated provisioning code	Generates and installs the public/private key pair into secure storage. Signs the certificate signing request that is sent to the CA.
Secure Storage	Infineon TPM	Storage on the IoT device that is designed to be protected from unauthorized access and capable of detecting attempts to hack or modify its contents. Used to generate, store, and process private keys, credentials, and other information that must be kept confidential.
General Factory Provisioning Instructions	Infineon TPM-specific Factory Provisioning Code	Creates a CSR associated with the key pair, installs the signed certificate into secure storage. Creates a record of devices that it has created and their certificates.
Build-specific Factory Provisioning Instructions	Build-specific Factory Provisioning Code	Sends device ownership information and the signed certificate to the MASA.
Manufacturer Database	MASA	When devices are manufactured, device identity and bootstrapping information is stored here by the manufacturer. Eventually, this database makes the device's bootstrapping information available to the device owner. Device bootstrapping information is information that the device owner requires to perform trusted network-layer onboarding; for BRSKI, the bootstrapping information is a signed certificate that is sent to the MASA, along with information regarding the device's owner.
Certificate Authority (CA)	SEALSQ INeS CMS CA NquiringMinds On-premises CA	Issues and signs certificates as needed.

H.3.2 BRSKI Factory Provisioning Build Logical Architectures

Figure H-1 depicts the logical architecture of the first version of the BRSKI factory provisioning build (i.e., the NquiringMinds and SEALSQ implementation) and is annotated with the steps that are performed in this build to prepare IoT devices for network-layer onboarding using the BRSKI protocol. Figure H-1 shows a Raspberry Pi device with a SEALSQ VaultIC SE attached. An SD card that contains factory provisioning code provided by SEALSQ and NquiringMinds is also required. To perform factory provisioning using this build, insert the SD card into the Raspberry Pi, as depicted (or activate the code in the factory provisioning partition of the SD card that is already in the Raspberry Pi). The SEALSQ software will boot up and perform the following steps to simulate the activities of a factory:

1. Instruct the SE to generate and store a private/public key pair
2. Create a certificate signing request for this key pair and have the SE sign it
3. Send the signed CSR to the IDevID CA (i.e., to the INeS CA that is operated by SEALSQ)
4. Receive back the signed certificate from the CA
5. Load the certificate into the SE
6. Send the certificate (along with device ownership information) to the manufacturer's database, which in this case is the MASA that is trusted by the owner

This completes the steps performed as part of the first version of the BRSKI Factory Provisioning Build. Once complete, shipment of the device to its owner can be simulated by walking the device across the room in the NCCoE laboratory to the Build 5 (NquiringMinds) implementation and replacing the SD card that has the factory provisioning code on it with and SD card that has the BRSKI onboarding code on it. (Alternatively, if the factory provisioning code and the BRSKI onboarding code are stored in separate partitions of the same SD card, shipment of the device to its owner can be simulated by booting up the code in the onboarding partition.) Build 5 is designed to execute this BRSKI onboarding software, which onboards the device to the device owner's network by provisioning the device with an LDevID that will serve as its network-layer credential. Such successful network-layer onboarding of the newly provisioned device using the BRSKI protocol by Build 5 would serve to confirm that the first version of the BRSKI factory provisioning process successfully provisioned the device with its birth credentials. At the time of this writing, however, this confirmation process was not able to be performed. In order to securely network-layer onboard the newly provisioned Raspberry Pi using the BRSKI protocol, the Raspberry Pi's onboarding software would need to be written to use the private key stored in the SEALSQ secure element when running the BRSKI protocol. Such software was not yet available at the time of this publication. The BRSKI onboarding code on the Raspberry Pi does not currently use the private key stored in the SEALSQ SE. As a result, Build 5 was not able to onboard this factory Pi as a way of confirming that the first version of the BRSKI factory build process completed successfully. The repository that hosts the code for this implementation can be found here at the trustnetz-se Github repository.

Figure H-1 Logical Architecture of the First Version of the BRSKI Factory Provisioning Build

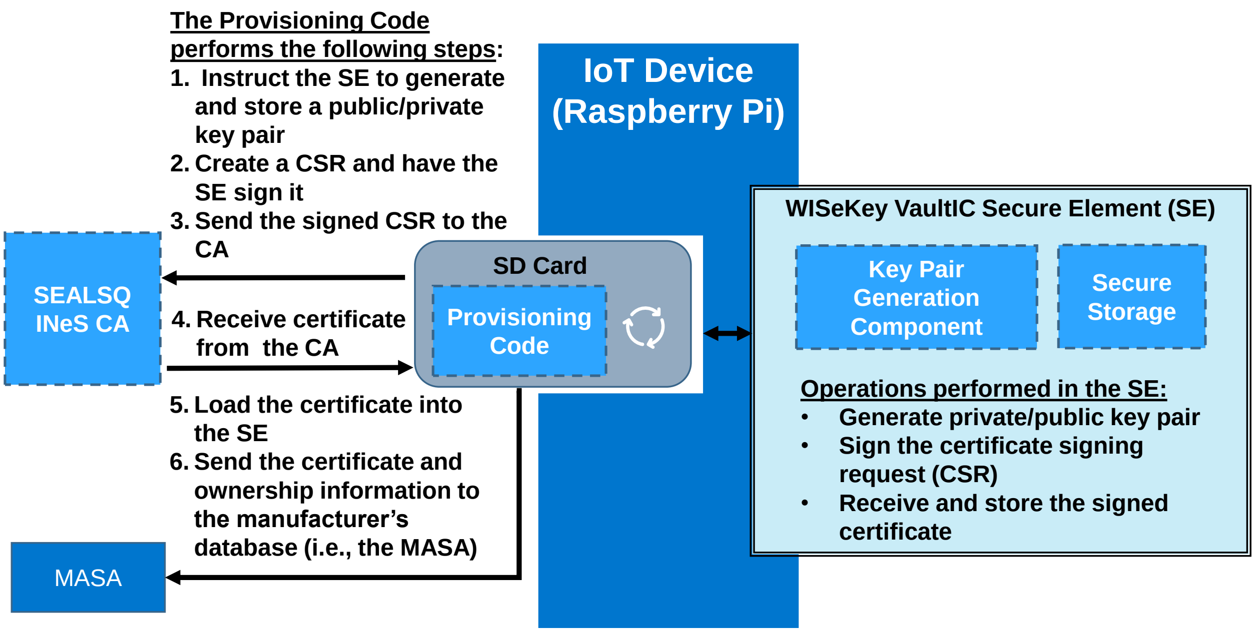
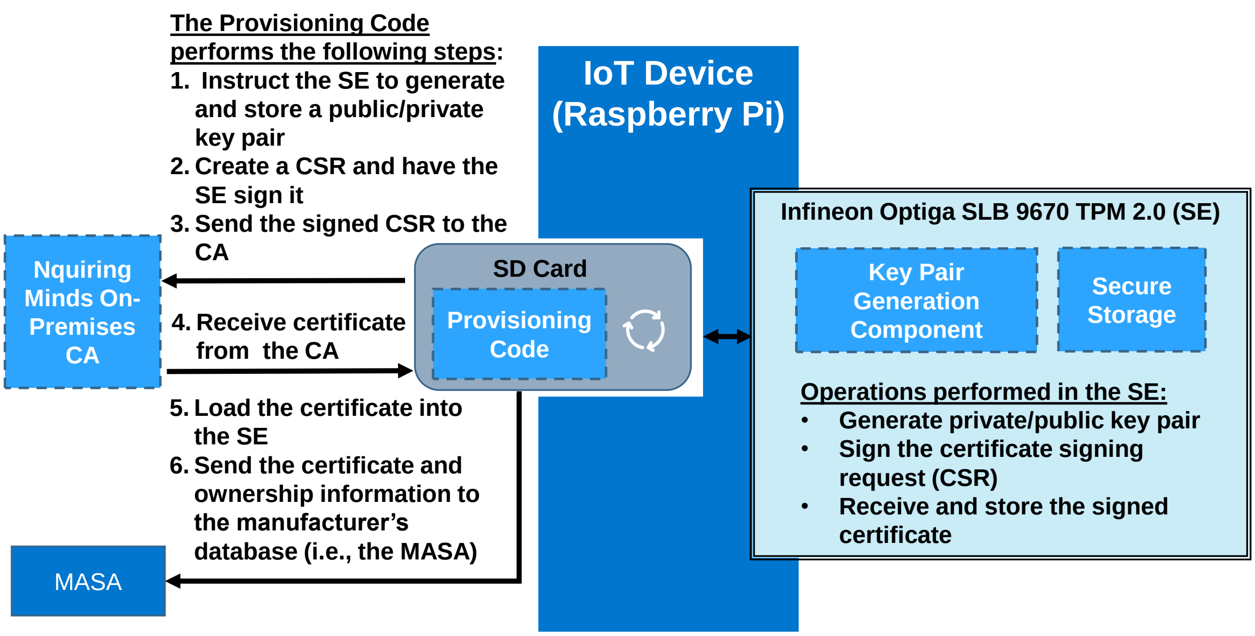


Figure H-2 depicts the logical architecture of the second version of the BRSKI factory provisioning build and is annotated with the steps that are performed in this build to prepare IoT devices for network-layer onboarding using the BRSKI protocol. Figure H-2 shows a Raspberry Pi device with an Infineon Optiga SLB 9670 TPM 2.0 SE attached. An SD card that contains factory provisioning code provided by NquiringMinds is also required. To perform factory provisioning using this build, insert the SD card into the Raspberry Pi, as depicted (or activate the code in the factory provisioning partition of the SD card that is already in the Raspberry Pi). The factory provisioning code software will boot up and perform the following steps to simulate the activities of a factory:

1. Instruct the Infineon SE to generate and store a private/public key pair
2. Create a certificate signing request for this key pair and have the SE sign it
3. Send the signed CSR to the IDevID CA (i.e., to the NquiringMinds on-premises CA/Manufacturer Provisioning Root)
4. Receive back the signed certificate from the CA
5. Load the certificate into the SE
6. Send the certificate (along with device ownership information) to the manufacturer's database, which in this case is the MASA that is trusted by the owner

This completes the steps performed as part of the second version of the BRSKI Factory Provisioning Build. Once complete, shipment of the device to its owner can be simulated by walking the device across the room in the NCCoE laboratory to the Build 5 (NquiringMinds) implementation and replacing the SD card that has the factory provisioning code on it with and SD card that has the BRSKI onboarding code on it. (Alternatively, if the factory provisioning code and the BRSKI onboarding code are stored in separate partitions of the same SD card, shipment of the device to its owner can be simulated by booting up the code in the onboarding partition.) Build 5 executes a modification of the BRSKI onboarding software that has been modified to use the IDevID resident on the Infineon TPM throughout the protocol flow, ensuring the device's IDevID's private key is never made public and never leaves the secure element. Specifically, the critical signing operations and the TLS negotiation steps are fully secured by the SE. The full BRSKI onboarding flow provisions a new LDevID onto the device. This LDevID provides the secure method for the device to connect to the domain owner's network. This successful network-layer onboarding of the IoT device by Build 5 serves as confirmation that the second version of the BRSKI factory provisioning process successfully provisioned the device with its birth credentials.

Figure H-2 Logical Architecture of the Second Version of the BRSKI Factory Provisioning Build



H.3.3 BRSKI Factory Provisioning Build Physical Architectures

Section 5.6.1 describes the physical architecture of the BRSKI Factory Provisioning Builds.

H.4 Wi-Fi Easy Connect Factory Provisioning Build (SEALSQ and Aruba/HPE)

SEALSQ, a subsidiary of WISeKey, and Aruba/HPE implemented a Wi-Fi Easy Connect Factory Provisioning Build. This build is designed to provision birth credentials to a Raspberry Pi device that has an attached secure element provided by SEALSQ.

H.4.1 Wi-Fi Easy Connect Factory Provisioning Build Technologies

The general infrastructure for the Wi-Fi Easy Connect Factory Provisioning Build is provided by SEALSQ. The Wi-Fi Easy Connect Factory Provisioning Build infrastructure consists of:

- A SEALSQ VaultIC SE that is attached to the Raspberry Pi

- SEALSQ Factory Provisioning Code that is located on an SD card and that communicates with the chip in the SE to:
    - create a P-256 Elliptic Curve public/private key pair within the SE,
    - use the public key to construct a DPP URI
    - export the DPP URI and convert it into a QR code

Table H-3 lists the technologies used in the Wi-Fi Easy Connect Factory Provisioning Build. It lists the products used to instantiate each logical build component and the security function that the component provides. The components listed are logical. They may be combined in physical form, e.g., a single piece of hardware may both generate key pairs and provide secure storage.

Table H-3 Wi-Fi Easy Connect Factory Provisioning Build Products and Technologies

Component	Product	Function
Key Pair Generation Component	SEALSQ VaultIC and associated provisioning code	Generates and installs the public/private key pair into secure storage. The VaultIC has a SP800-90B certified random number generator for key pair generation. [17]
Secure Storage	SEALSQ VaultIC	Storage on the IoT device that is designed to be protected from unauthorized access and capable of detecting attempts to hack or modify its contents. Used to generate, store, and process private keys, credentials, and other information that must be kept confidential.
General Factory Provisioning Instructions	SEALSQ Factory Provisioning Code	Creates a public/private key pair.
Build-specific Factory Provisioning Instructions	Aruba/HPE Factory Provisioning Code	Uses the public key to create a DPP URI. Exports the DPP URI and converts it into a QR code.
Manufacturer Database	Manufacturer cloud or imprint on device	The DPP URI information is stored in the QR code and is the mechanism for conveying the device's bootstrapping information to the device owner.

H.4.2 Wi-Fi Easy Connect Factory Provisioning Build Logical Architecture

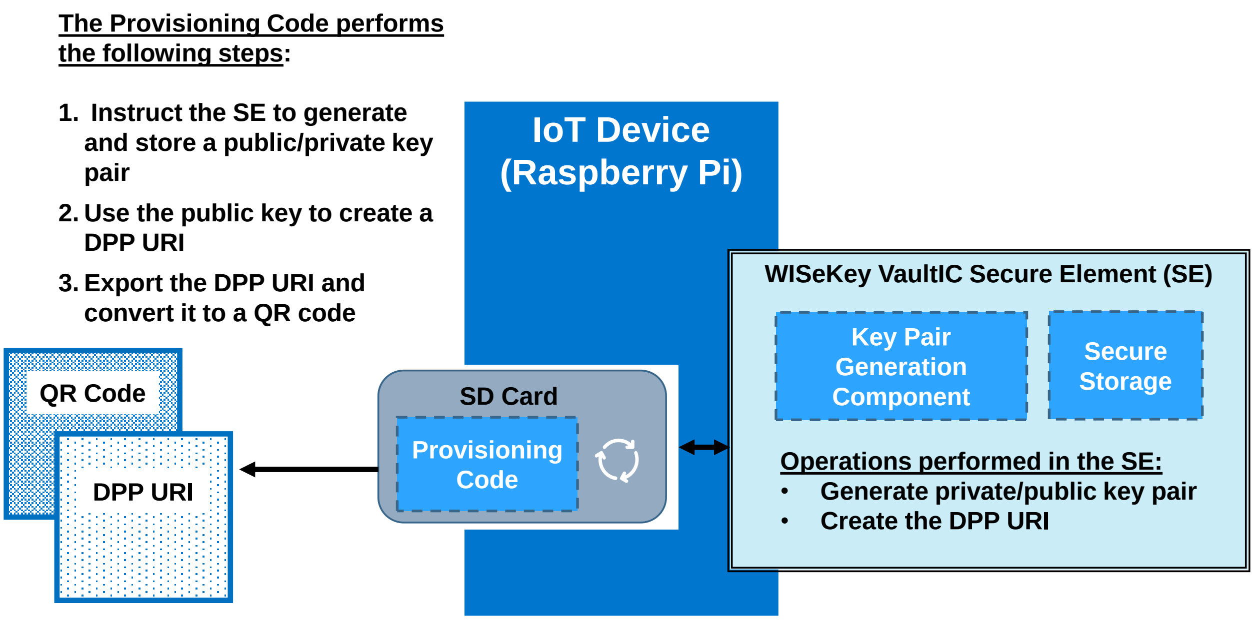
Figure H-3 depicts the logical architecture of the Wi-Fi Easy Connect factory provisioning build and is annotated with the steps that are performed in this build to prepare Raspberry Pi IoT devices for network-layer onboarding using the Wi-Fi Easy Connect protocol. Figure H-3 shows a Raspberry Pi device with a SEALSQ VaultIC SE attached. Factory provisioning code provided by SEALSQ and Aruba/HPE must also be loaded. In Figure H-3, this code is shown as being on an SD card. The factory provisioning software will boot up and perform the following steps to simulate the activities of a factory:

1. Instruct the SE to generate and store a private/public key pair
2. Use the public key to create a DPP URI

3. Export the DPP URI and convert it into a QR code

This completes the steps performed as part of the Wi-Fi Easy Connect Factory Provisioning Build. Once complete, shipment of the device to its owner can be simulated by walking the device across the room in the NCCoE laboratory to the Build 1 (Aruba/HPE) implementation. Build 1 uses the Wi-Fi Easy Connect protocol to network-layer onboard the device to the device owner's network by provisioning the device with connector that will serve as its network-layer credential. Successful network-layer onboarding of the newly provisioned device using the Wi-Fi Easy Connect protocol by Build 1 would serve to confirm that the Wi-Fi Easy Connect factory provisioning process correctly provisioned the device with its birth credentials. At the time of this writing, however, this confirmation process was not able to be performed. In order to securely network-layer onboard the newly provisioned Raspberry Pi using the Wi-Fi Easy Connect protocol, the Raspberry Pi would need to be equipped with a firmware image that uses the private key stored in the secure element when running the Wi-Fi Easy Connect protocol. Such firmware was not yet available at the time of this publication. The Wi-Fi Easy Connect code on the Raspberry Pi does not use the private key stored in the SE at this time. Confirmation that the factory build process completed successfully is limited to inspection of the .PNG file and .URI file that were created to display the QR Code and the device's DPP URI, respectively.

Figure H-3 Logical Architecture of the Wi-Fi Easy Connect Factory Provisioning Build



H.4.3 Wi-Fi Easy Connect Factory Provisioning Build Physical Architecture

Section 5.2.1 describes the physical architecture of the Factory Provisioning Build.

Appendix I References

[1] L. S. Vailshery, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030," Statista, July 2023. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[2] S. Symington, W. Polk, and M. Souppaya, *Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management (Draft)*, National Institute of Standards and Technology (NIST) Draft Cybersecurity White Paper, Gaithersburg, MD, Sept. 2020, 88 pp. https://doi.org/10.6028/NIST.CSWP.09082020-draft.

[3] E. Lear, R. Droms, and D. Romascanu, *Manufacturer Usage Description Specification,* IETF Request for Comments (RFC) 8520, March 2019. Available: https://tools.ietf.org/html/rfc8520.

[4] M. Souppaya et al, *Securing Small-Business and Home Internet of Things (IoT) Devices: Mitigating Network-Based Attacks Using Manufacturer Usage Description (MUD)*, National Institute of Standards and Technology (NIST) Special Publication (SP) 1800-15, Gaithersburg, Md., May 2021, 438 pp. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-15.pdf.

[5] "National Cybersecurity Center of Excellence (NCCoE) Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management," Federal Register Vol. 86, No. 204, October 26, 2021, pp. 59149-59152. Available: https://www.federalregister.gov/documents/2021/10/26/2021-23293/national-cybersecurity-center-of-excellence-nccoe-trusted-internet-of-things-iot-device.

[6] Wi-Fi Alliance, *Wi-Fi Easy Connect™ Specification Version 3.0*, 2022. Available: https://www.wi-fi.org/system/files/Wi-Fi_Easy_Connect_Specification_v3.0.pdf.

[7] M. Pritikin, M. Richardson, T.T.E. Eckert, M.H. Behringer, and K.W. Watsen, *Bootstrapping Remote Secure Key Infrastructure (BRSKI)*, IETF Request for Comments (RFC) 8995, October 2021. Available: https://datatracker.ietf.org/doc/rfc8995/.

[8] Thread 1.1.1 Specification, February 13, 2017.

[9] OpenThread Released by Google. Available: https://openthread.io/.

[10] O. Friel, E. Lear, M. Pritikin, and M. Richardson, *BRSKI over IEEE 802.11*, IETF Internet-Draft (Individual), July 2018. Available: https://datatracker.ietf.org/doc/draft-friel-brski-over-802dot11/01/.

[11] NIST. *The NIST Cybersecurity Framework (CSF) 2.0*. Available: https://doi.org/10.6028/NIST.CSWP.29.

[12] *IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity*, IEEE Std 802.1AR-2018 (Revision of IEEE Std 802.1AR-2009), 2 Aug. 2018, 73 pp. Available: https://ieeexplore.ieee.org/document/8423794.

[13] F. Stajano and R. Anderson, *The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks*, B. Christianson, B. Crispo and M. Roe (Eds.). Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science, 1999. Springer-Verlag Berlin Heidelberg 1999. Available: https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf.

[14] M. Richardson, *A Taxonomy of operational security considerations for manufacturer installed keys and Trust Anchors*, IETF Internet-Draft (Individual), November 2022. Available: https://datatracker.ietf.org/doc/draft-richardson-t2trg-idevid-considerations/.

[15] Certificate #4302, Cryptographic Module Validation Program, NIST Computer Security Resource Center. Available: https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4302.

[16] Certificate #4303, Cryptographic Module Validation Program, NIST Computer Security Resource Center. Available: https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4303.

[17] Entropy Certificate #E2, Cryptographic Module Validation Program, NIST Computer Security Resource Center. Available: https://csrc.nist.gov/projects/cryptographic-module-validation-program/entropy-validations/certificate/2.