Network Working Group
Internet-Draft
Intended status: Informational
Expires: 22 April 2026

M. Sethi
Aalto University
B. Sarikaya

D. Garcia-Carrillo
University of Oviedo
19 October 2025

Terminology and processes for initial security setup of IoT devices
draft-irtf-t2trg-security-setup-iot-devices-05

Abstract

This document provides an overview of terms that are commonly used when discussing the initial security setup of Internet of Things (IoT) devices. This document also presents a brief but illustrative survey of protocols and standards available for initial security setup of IoT devices. For each protocol, we identify the terminology used, the entities involved, the initial assumptions, the processes necessary for completion, and the knowledge imparted to the IoT devices after the setup is complete.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 April 2026.

Copyright Notice

Table of Contents

1. Introduction 2
2. Standards and Protocols 4
2.1. Bluetooth 4
2.2. Device Provisioning Protocol (DPP) 6
2.3. Enrollment over Secure Transport (EST) 7
2.4. Open Mobile Alliance (OMA) Lightweight Machine to Machine specification (LwM2M) 9
2.5. Nimble out-of-band authentication for EAP (EAP-NOOB) . . 12
2.6. Open Connectivity Foundation (OCF) 13
2.7. Bootstrapping Remote Secure Key Infrastructures (BRSKI) 14
2.8. Secure Zero Touch Provisioning (SZTP) 15
2.9. FIDO Device Onboard (FDO) 18
2.10. LPWAN 22
2.11. Thread 23
3. Comparison 25
3.1. Comparison of terminology 25
3.2. Comparison of players 28
3.3. Comparison of initial beliefs 28
3.3.1. No initial trust established 29
3.3.2. Initial trust based on the credentials installed . . 29
3.4. Comparison of processes 30
3.5. Comparison of knowledge imparted to the device 30
4. Security Considerations 31
5. IANA Considerations 32
6. Acknowledgements 32
7. Informative References 32
Authors' Addresses 35

1. Introduction

Initial security setup for a device refers to any process that takes place before a device can become fully operational. The phrase *initial security setup* intentionally includes the term _security_ as setup of devices without adequate security or with insecure processes is no longer acceptable. The initial security setup process, among other things, involves network discovery and selection, access authentication, and configuration of necessary credentials and parameters.

Initial security setup for IoT devices is challenging because the size of an IoT network varies from a couple of devices to tens of thousands, depending on the application. Moreover, devices in IoT networks are produced by a variety of vendors and are typically heterogeneous in terms of the constraints on their power supply, communication capability, computation capacity, and user interfaces available. This challenge of initial security setup in IoT was identified by Sethi et al. [Sethi14] while developing a solution for smart displays.

Initial security setup of devices typically also involves providing them with some sort of network connectivity. The functionality of a disconnected device is rather limited. Initial security setup of devices often assumes that some network has been setup a-priori. Setting up and maintaining a network itself is challenging. For example, users may need to configure the network name (called as Service Set Identifier (SSID) in Wi-Fi networks) and passphrase before new devices can be setup. Specifications such as the Wi-Fi Alliance Simple Configuration [simpleconn] help users setup networks. However, this document is only focused on terminology and processes associated with the initial security setup of devices and excludes any discussion on setting up networks.

There are several terms that are used in the context of initial security setup of devices:

* Bootstrapping

* Provisioning

* Onboarding

* Enrollment

* Commissioning

* Initialization

* Configuration

* Registration

* Discovery

In addition to using a variety of different terms, initial security setup mechanisms can rely on several entities. For example, a companion smartphone device may be necessary for some initial security setup mechanisms. Moreover, security setup procedures have

diverse initial assumptions about the device being setup. As an example, an initial security setup mechanism may assume that the device is provisioned with a pre-shared key and a list of trusted network identifiers. Finally, initial security setup mechanisms impart different information to the device after completion. For example, some mechanisms may only provide a key for use with an authorization server, while others may configure elaborate access control lists directly.

The next section provides an overview of some standards and protocols for initial security setup of IoT devices. For each mechanism, the following are explicitly identified:

* Terminology used

* Entities or "players" involved

* Initial assumptions about the device

* Processes necessary for completion

* Knowledge imparted to the device after completion

2. Standards and Protocols

2.1. Bluetooth

Bluetooth mesh specifies a provisioning protocol. The goal of the provisioning phase is to securely incorporate a new Bluetooth mesh node, by completing two critical tasks. First, to authenticate the unprovisioned device and second, to create a secure link with said device to share information.

The provisioning process is divided into five distinct stages summarize next:

* Beaconing for discovery: The new unprovisioned device is discovered by the provisioner

* Negotiation: The unprovisioned device indicates to the provisioner a set of capabilities such as the security algorithms supported, the availability of its public key using an Out-of-Band (OOB) channel and the input/output interfaces supported

* Public-key exchange: The authentication method is selected by the provisioner and both devices exchange Elliptic-curve Diffie-Hellman (ECDH) public keys. These keys may be static or ephemeral. Their exchange can be done in two ways, either via

Out-of-Band or directly through a Bluetooth link. Each device then generates a symmetric key, named ECDHSecret, by combining its own private key and the public key of the peer device. The EDCHSecret is used to protect communication between the two devices.

* Authentication: During this phase, the ECDH key exchange is authenticated. The authentication method can be Output OOB, Input OOB, static OOB, or No OOB. With Output OOB, the unprovisioned device chooses a random number and outputs that number in manner consistent with its capabilities. The provisioner then inputs this number. Then, a check confirmation value operation is performed. This involves a cryptographic exchange regarding (in this case) the random number to complete the authentication. With Input OOB, the roles are reversed, being the provisioner the entity that generates the random number. When neither of the previous authentication procedures are feasible, both the provisioner and unprovisioned device generate a random number and require the user supervising the setup to verify that values on the device and provisioner are the same.

* Distribution of provisioning data: At this point, the provisioning process can be protected. This involves the distribution of data such as a Network key, to secure the communications at network layer and a unicast address among other information.

Bluetooth mesh has the following characteristics:

* Terms: Configuration, discovery, provisioning.

* Players: Client, Device, Manager, Manufacturer, Peer, Server, User

* Initial beliefs assumed in the device: Previously to the provisioning phase, there are no pre-shared credentials for a trust relation.

* Processes: Provisioning

* Beliefs imparted to the device after protocol execution: After the provisioning, the device trusts the provisioner entity and any other device in the network sharing its network key.

2.2. Device Provisioning Protocol (DPP)

The Wi-Fi Alliance Device provisioning protocol (DPP) [dpp] describes itself as a standardized protocol for providing user-friendly Wi-Fi setup while maintaining or increasing the security. DPP relies on a configurator, e.g. a smartphone application, for setting up all other devices, called enrollees, in the network. DPP has the following three phases/sub-protocols:

* Bootstrapping: The configurator obtains bootstrapping information from the enrollee using an out-of-band channel such as scanning a QR code or tapping NFC. The bootstrapping information includes the public-key of the device and metadata such as the radio channel on which the device is listening.

* Authentication: In DPP, either the configurator or the enrollee can initiate the authentication protocol. The side initiating the authentication protocol is called the initiator while the other side is called the responder. The authentication sub-protocol provides authentication of the responder to an initiator. It can optionally authenticate the initiator to the responder (only if the bootstrapping information was exchanged out-of-band in both directions).

* Configuration: Using the key established from the authentication protocol, the enrollee asks the configurator for network information such as the SSID and passphrase of the access point.

DPP has the following characteristics:

* Terms: Bootstrapping, configuration, discovery, enrollment, provisioning.

* Players: Authenticator, Client, Configurator, Device, Initiator, Manufacturer, Owner, Peer, Persona, Responder, User, Enrollee

* Initial beliefs assumed in the device: There are two entities involved in the DPP protocol, the Initiator and Responder. These entities as a starting point do not have a trust relation, nor do they share credentials or key material. DPP uses a decentralized architecture with no central authority to coordinate or control authentication and rely on a direct trust model. In DPP, authentication does not rely on a pre-existing trust relation with a third-party or centralized entity, hence all entities involved in DPP need to perform the required validation.

* Processes: Bootstrapping, authentication, provisioning, and network access. Bootstrapping: To establish a secure provisioning connection, the devices exchange public bootstrapping keys. Authentication: To establish trust and build a secure channel, the devices employ the DPP Authentication protocol’s bootstrapping keys. Configuration: The Configurator uses the DPP Configuration protocol to provision the Enrollee through the secure channel created during DPP Authentication. Network access: The Enrollee establishes network access using the newly provisioned credentials.

* Beliefs imparted to the device after protocol execution: DPP bootstrapping relies on the transfer of the public-key that is expected to be trusted. The beliefs when mutual authentication is run, relies on the trust of a successful DPP bootstrapping. When mutual authentication is not supported, a device that can control when and how its own public key is bootstrapped, can perform a weak authentication to any entity that knows its public key.

2.3. Enrollment over Secure Transport (EST)

Enrollment over Secure Transport (EST) [RFC7030] defines a profile of Certificate Management over CMS (CMC) [RFC5272]. EST relies on Hypertext Transfer Protocol (HTTP) and Transport Layer Security (TLS) for exchanging CMC messages and allows client devices to obtain client certificates and associated Certification Authority (CA) certificates. A companion specification for using EST over secure CoAP has also been standardized [RFC9148]. EST assumes that the enrolling device already has IP connectivity to the EST server and some initial information is already distributed so that EST client and server to perform mutual authentication before continuing with protocol. [RFC7030] further defines "Bootstrap Distribution of CA Certificates" which allows minimally configured EST clients to obtain initial trust anchors. It relies on human users to verify information such as the CA certificate "fingerprint" received over the unauthenticated TLS connection setup. After successful completion of this bootstrapping step, clients can proceed to the enrollment step during which they obtain client certificates and associated CA certificates.

EST has the following characteristics:

* *Terms*:

  - _Bootstrapping_: used for the process when the client device has not been configured with an Implicit Trust Anchor (TA) database and device owner or administrator manually verifies the fingerprint of the EST CA certificate. This manual

verification is only needed once, as the device establishes an explicit TA database for subsequent TLS authentication of the EST server.

- _Enrollment_: describes the entire process of obtaining a client certificate from the EST CA. This includes learning the EST CA certificate, discovering required attributes for the Certificate Signing Request (CSR), submitting the CSR, and receiving the final client certificate.

- _Initialization_: term used to refer to the essential initialization data that the device needs for completing the enrollment including the trust anchors, the EST server URI, and credentials for TLS authentication.

* *Players*: The network administrator is responsible for deploying and maintaining the EST CA and EST server before a device can be enrolled into the network. The device manufacturer must install a client identity certificate (such as an IEEE 802.1AR certificate), a shared secret for TLS authentication without certificates, and/or a username and password for HTTP Basic or Digest authentication. Additionally, the manufacturer may configure trust anchors for verifying the EST server and set the EST server URI. However, the EST specification allows for manual configuration of all required information, including the manual verification of the EST CA certificate fingerprint.

* *Initial beliefs assumed in the device*: A device acting as an EST client is assumed to possess an existing credential for authenticating itself during the TLS handshake with the EST server. This credential may be a previously issued EST client certificate or a certificate from a distinct PKI, such as an IEEE 802.1AR manufacturer-installed certificate. Alternatively, the client may authenticate using a shared secret-based credential, such as a pre-installed symmetric key, or a username and password, either as a standalone method or in combination with TLS authentication. To verify the EST server, the client relies on a trust anchor database, which may be explicit, used to authenticate certificates issued by the EST CA, including the EST server certificate, or implicit, allowing authentication of the EST server when it presents a certificate issued by an external CA. The implicit trust anchor database may initially contain pre-installed CA certificates and can be disabled once the EST CA certificate is obtained. The EST client must also be configured with a Uniform Resource Identifier (URI) to locate the EST server.

* *Processes*: Before a device can enroll using EST, the necessary network infrastructure must be in place, including the deployment of the EST server and CA. The device can discover the EST server URI through various methods, such as manual configuration or preconfigured settings from the manufacturer. If the device lacks an explicit Trust Anchor (TA) database, it may require bootstrapping, where the owner or administrator manually verifies the fingerprint of the EST CA certificate. Once the EST server is authenticated, the device retrieves the EST CA certificate, learns the required attributes for generating a Certificate Signing Request (CSR), and submits the CSR to obtain a client certificate. After enrollment, the device can securely authenticate to the network using its newly issued certificate and renew it before expiration.

* *Beliefs imparted to the device after protocol execution*: After completing the enrollment process, the device obtains a client certificate issued by the EST CA, which it can use for authentication in subsequent communications. During enrollment, the device also learns the EST CA certificate, along with any intermediate certificates needed to build a complete trust chain to the EST CA trust anchor. The client also discovers the attributes it should include in a Certificate Signing Request (CSR), such as key usage, extended key usage, etc. Depending on the enrollment method, the device may generate its own key pair and submit a CSR or receive both a server-generated key pair and certificate. If the client requested a Full PKI, it may also receive information such as certificate revocation lists (CRLs), policy and name constriants etc. as defined in [RFC5272].

2.4. Open Mobile Alliance (OMA) Lightweight Machine to Machine specification (LwM2M)

LwM2M specification developed by OMA [oma] defines a RESTful architecture where a new IoT device (LwM2M client) first contacts an LwM2M Bootstrap-Server for obtaining essential information such credentials for subsequently registering with one or more LwM2M Servers. These one or more LwM2M servers are used for performing device management actions during the device lifecycle (reading sensor data, controlling an actuator, modifying access controls etc.). LwM2M specification does not deal with the initial network configuration of IoT devices and assumes that the IoT client device has network reachability to the LwM2M Bootstrap-Server and LwM2M Server.

The current standard defines the following four bootstrapping modes:

* Factory Bootstrap: An IoT device is configured with all the information necessary for securely communicating with an LwM2M Bootstrap-Server and/or LwM2M Server while it is manufactured and prior to its deployment.

* Bootstrap from Smartcard: An IoT device retrieves all the information necessary for securely communicating with an LwM2M Bootstrap-Server and/or LwM2M Server from a Smartcard.

* Client Initiated Bootstrap: If the IoT device in one of the above bootstrapping modes is only configured with information about an LwM2M Bootstrap-Server, then the client device must first communicate securely with the configured LwM2M Bootstrap-Server and obtain the necessary information and credentials to subsequently register with one or more LwM2M Servers.

* Server Initiated Bootstrap: In this bootstrapping mode, the LwM2M server triggers the client device to begin the client initiated bootstrap sequence described above.

The LwM2M specification is also quite flexible in terms of the credentials and the transport security mechanism used between the client device and the LwM2M Server or the LwM2M Bootstrap-Server. Credentials such as a pre-shared symmetric key, a raw public key (RPK), or X.509 certificates can be used with various transport protocols such as Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS) as specified in LwM2M transport bindings specification [oma-transport].

As explained earlier, an LwM2M Bootstrap-Server is responsible for provisioning credentials into an LwM2M Client. When X.509 certificates are being provisioned, the asymmetric key pair is generated on the Bootstrap-Server and then sent to the LwM2M client device. This approach is not acceptable in all scenarios and therefore, LwM2M specification also supports a mode where the client device uses the Enrollment over Secure Transport (EST) certificate management protocol for provisioning certificates from the LwM2M Bootstrap-Server to the LwM2M Client.

OMA has the following characteristics:

* *Terms*:

- _Bootstrapping_ and _Unbootstrapping_: Bootstrapping is used for describing the process of providing an IoT device with credentials and information of one or more LwM2M servers. Interestingly, the transport bindings specification [oma-transport] also uses the term unbootstrapping for the process where objects corresponding to an LwM2M Server are deleted on the client.

- _Provisioning_ and _configuration_: terms used to refer to the process of providing some information to a LwM2M client.

- _Discovery_: term for the process by which a LwM2M Bootstrap-Server or LwM2M Server discovers objects, object instances, resources, and attributes supported by RESTful interfaces of a LwM2M Client.

- _Register_ and _De-register_: Register is the process by which a client device sets up a secure association with an LwM2M Server and provides the server with information about objects and existing object instances of the client. De-register is the process by which the client deletes information about itself provided to the LwM2M server during the registration process.

- _Intialization_: term for the process by which an LwM2M Bootstrap-Server or LwM2M Server deletes objects on the client before performing any write operations.

* *Players*: Device manufacturers or Smartcard manufacturers are responsible for providing client IoT devices with initial information and credentials of LwM2M Bootstrap-Server and/or LwM2M server.

* *Initial beliefs assumed in the device*: The client at the very least has the necessary information to trust the LwM2M bootstrap server.

* *Processes*: LwM2M does not require any actions from the device owner/user. Once the device is registered with the LwM2M server, various actions related to device management can be performed by device owner/user via the LwM2M server.

* *Beliefs imparted to the device after protocol execution*: After the bootstrapping is performed, the LwM2M client can register (Security object and Server object) with the LwM2M servers.

2.5. Nimble out-of-band authentication for EAP (EAP-NOOB)

Extensible Authentication Protocol (EAP) framework provides support for multiple authentication methods. EAP-NOOB [RFC9140] defines an EAP method where the authentication is based on a user-assisted out-of-band (OOB) channel between the IoT device (peer in EAP terminology) and the server. It is intended as a generic bootstrapping solution for IoT devices which have no pre-configured authentication credentials and which are not yet registered on the authentication server.

The application server where the IoT device is registered once EAP-NOOB is completed may belong to the manufacturer or the local network where the device is being deployed. EAP-NOOB uses the flexibility of the Authentication, Authorization, and Accounting (AAA) [RFC2904] architecture to allow routing of EAP-NOOB sessions to a specific application server.

EAP-NOOB claims to be more generic than most ad-hoc bootstrapping solutions in that it supports many types of OOB channels and supports IoT devices with only output (e.g. display) or only input (e.g. camera).

EAP-NOOB has the following characteristics:

* *Terms*:
  - _Bootstrapping_: used to describe the entire process involved during the initial security setup of an IoT device. The specification does not use separate terms or distinguish the process of obtaining identifier and credentials for communicating with an application server where the user has an account or for network connectivity.
  - _Registration_: describes the process of associating the device with a user account on an application server.
* *Players*: The device owner/user is responsible for transferring an OOB message necessary for protocol completion. The application server where the device is registered may be provided by different service providers including the device manufacturer or device owner. The local network needs standard AAA configuration for routing EAP-NOOB sessions to the application server chosen by the device owner/user.

* *Initial beliefs assumed in the device*: EAP-NOOB does not require devices to have any pre-installed credentials but expects all devices to use a standard identifier (noob@eap-noob.arpa) during initial network discovery.

* *Processes*: The IoT device performs network discovery and one or more OOB outputs may be generated. The user is expected EAP exchange is encompassed by Initial Exchange, OOB step, Completion Exchange and Waiting Exchange.

* *Beliefs imparted to the device after protocol execution*: After EAP-NOOB bootstrapping process is complete, the device and server establish a long-term secret, which can be renewed without further user involvement. As a side-effect, the device also obtains identifier and credentials for network and Internet connectivity (via the EAP authenticator).

2.6. Open Connectivity Foundation (OCF)

The Open Connectivity Foundation (OCF) [ocf] defines the process before a device is operational as onboarding. The first step of this onboarding process is configuring the ownership, i.e., establishing a legitimate user that owns the device. For this, the user is supposed to use an Onboarding tool (OBT) and an Owner Transfer Method (OTM).

The OBT is described as a logical entity that may be implemented on a single or multiple entities such as a home gateway, a device management tool, etc. OCF lists several optional OTMs. At the end of the execution of an OTM, the onboarding tool must have provisioned an Owner Credential onto the device. The following owner transfer methods are specified:

* Just works: Performs an un-authenticated Diffie-Hellman key exchange over Datagram Transport Layer Security (DTLS). The key exchange results in a symmetric session key which is later used for provisioning. Naturally, this mode is vulnerable to on-path attackers.

* Random PIN: The device generates a PIN code that is entered into the onboarding tool by the user. This pin code is used together with TLS-PSK ciphersuites for establishing a symmetric session key. OCF recommends PIN codes to have an entropy of 40 bits.

* Manufacturer certificate: An onboarding tool authenticates the device by verifying a manufacturer-installed certificate. Similarly, the device may authenticate the onboarding tool by verifying its signature.

* Vendor specific: Vendors implement their own transfer method that accommodates any specific device constraints.

Once the onboarding tool and the new device have authenticated and established secure communication, the onboarding tool provisions/ configures the device with Owner credentials. Owner credentials may consist of certificates, shared keys, or Kerberos tickets for example.

The OBT additionally configures/provisions information about the Access Management Service (AMS), the Credential Management Service (CMS), and the credentials for interacting with them. The AMS is responsible for provisioning access control entries, while the CMS provisions security credentials necessary for device operation.

OCF has the following characteristics:

* Terms: Configuration, discovery, enrollment, onboarding, provisioning, registration,

* Players: Client, Device, Manager, Manufacturer, Owner, Peer, Server, User

* Initial beliefs assumed in the device: The device needs to be associated with an owner in the onboarding process and then go through the provisioning process before being considered as trustworthy.

* Processes: Onboarding, provisioning.

* Beliefs imparted to the device after protocol execution: In the provisioning phase the device receives the necessary credentials to interact with provisioning services and any other services or devices that are part of the normal operation of the device.

2.7. Bootstrapping Remote Secure Key Infrastructures (BRSKI)

The ANIMA working group is working on a bootstrapping solution for devices that rely on 802.1AR [ieee8021ar] vendor certificates called Bootstrapping Remote Secure Key Infrastructures (BRSKI) [RFC8995]. In addition to vendor installed IEEE 802.1AR certificates, a vendor based service on the Internet is required. Before being authenticated, a new device only needs link-local connectivity, and does not require a routable address. When a vendor provides an Internet based service, devices can be forced to join only specific domains. The document highlights that the described solution is aimed in general at non-constrained (i.e. class 2+ defined in [RFC7228]) devices operating in a non-challenged network. It claims

to scale to thousands of devices located in hostile environments, such as ISP provided CPE devices which are drop-shipped to the end user.

BRSKI has the following characteristics:

* Terms: Bootstrapping, provisioning, enrollment, onboarding.

* Players: Administrator, Client, Cloud Registrar, Configurator, Device, Domain Registrar, Initiator, Join Proxy, JRC, Manufacturer, Owner, Peer, Pledge, Server, User

* Initial beliefs assumed in the device: Every device has an IDevID, installed and signed by the manufacturer, which is used as a basis for establishing further trust relations. In the initial stage, when the device is deployed in a specific location it cannot securely communicate with the registrar or JRC, to be integrated into the network, so the device and the registrar need to establish mutual trust.

* Processes: Discover, self-Identify, joint registrar, imprint registrar, enroll.

* Beliefs imparted to the device after protocol execution: After the process has finished and the device is imprinted, and trusts the registrar/JRC, through a voucher issued by the manufacturer and verified by the device.

2.8. Secure Zero Touch Provisioning (SZTP)

[RFC8572] defines a bootstrapping strategy that enables devices to securely obtain all configuration information with minimal installer input, beyond physical placement and cable connections. The goal of this bootstrapping protocol is to enable a secure NETCONF [RFC6241] or RESTCONF [RFC8040] connection to the deployment-specific network management system (NMS). Devices receive signed and optionally encrypted information about the owner’s NMS, which they authenticate using an owner certificate and an ownership voucher [RFC8366] signed by the device manufacturer. The owner certificate and ownership voucher can be delivered to the device via Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), removable storage (e.g., USB drives), or knowledge of well-known bootstrapping servers. Devices may be redirected to multiple servers before acquiring the necessary credentials to verify and connect to the designated NMS.

SZTP has the following characteristics:

* *Terms*:

- _Bootstrapping_: used to describe the entire process involved during the initial security setup of devices. It involves the device authenticating itself with manufacturer-issued certificates, fetching the owner certificate and ownership voucher, and retrieving the signed or encrypted onboarding information, such as the boot image, initial configuration, and scripts.

- _Provisioning_: refers to the process of providing all the necessary bootstrap information to a device so that it can become functional. In some sense, the terms bootstrapping and provisioning are used interchangeably because, for provisioning the bootstrap information onto the device, it is necessary to bootstrap the device. In fact, the specification also refers to Secure Zero Touch Provisioning (SZTP) as a _bootstrapping strategy_. Interestingly, even though the title of the protocol specification includes the word provisioning, it is used much less than bootstrapping in the specification text.

- _Onboarding_: used to refer to the information that the device needs to join the owner’s network. In particular, the onboarding information includes the boot image the device must run, an initial configuration the device must use, and scripts that the device must successfully execute. Note that in SZTP, onboarding information is not the entire set of information that the device needs for bootstrapping. Prior to obtaining the onboarding information, the device also needs the owner certificate and ownership voucher to verify the onboarding information received later.

- _Enrollment_: describes the process by which the device owner registers with the device manufacturer, ensuring that the manufacturer recognizes the owner and issues the necessary ownership vouchers. This crucial association is later used by the manufacturer to provide the owner with the serial numbers of the devices based on the order. The owner then uses these serial numbers during the bootstrapping process to verify the devices as their own.

- _Discovery_: used for describing the process by which devices discover sources of information, particularly bootstrap servers that are not already known to the device. This discovery typically happens via redirects from trusted bootstrapping servers.

* *Players*: SZTP requires significant involvement of the device manufacturer. The manufacturer is responsible for installing the client identity certificate and trust anchors for verifying

botstrapping server certificates and ownership vouchers during the manufacturing process. Additionally, the device manufacturer must support the enrollment of the owner by verifying the owner certificate and issuing an ownership voucher. After receiving an order for devices from an enrolled owner, the manufacturer needs to inform the owner of the serial numbers corresponding to the ordered devices. This naturally necessitates robust asset tracking systems. Lastly, the manufacturer may also need to operate well-known bootstrapping servers that the device contacts to retrieve the owner certificate and ownership voucher. At the same time, the device owner also carries substantial responsibility, including enrolling in the manufacturer's SZTP program, managing the keys associated with the owner certificate, and maintaining the network infrastructure to authenticate the device's serial number and confirm its association with the order placed with the manufacturer. Once authenticated, the owner is responsible for sending signed and/or encrypted onboarding information to the device.

* *Initial beliefs assumed in the device*: SZTP requires devices to have a pre-configured state, including a client X.509 certificate for TLS client authentication to bootstrap servers. While the specification allows the use of HTTP authentication with passwords, it typically relies on X.509 certificates in the form of IDevID certificates, as defined in [ieee8021ar]. Additionally, devices must have a list of trusted bootstrap servers and trust anchor certificates for verifying bootstrap server certificates and ownership vouchers signed by the manufacturer. All this information, including the client TLS certificate and trust anchors, must be installed during manufacturing.

* *Processes*: A precursor for the device to bootstrap and join the owner's network is the enrollment of the owner with the manufacturer and the setup of all necessary network infrastructure to authenticate the device. Thereafter, the device can use various sources such as Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP) options, removable storage (e.g., USB drives), or knowledge of well-known bootstrapping servers to locate the owner certificate and ownership voucher. These methods can be used in parallel to expedite the process. Once the owner certificate and ownership voucher are obtained and verified, the device receives, verifies/decrypts the signed and/or encrypted onboarding information, including boot images, configuration details, and scripts. With the image, configuration, and scripts, the device can securely join the owner's network management system (NMS). Unlike other protocols, once all the infrastructure has been set up, no actions are needed on the device itself other than its physical placement, cabling, and booting up.

* *Beliefs imparted to the device after protocol execution*: After the SZTP bootstrapping process, the device possesses all necessary information, called as bootstrapping data, for secure communication with the deployment-specific NMS. This bootstrapping data includes the ownership voucher, the owner certificate, and onboarding information. The owner certificate and ownership voucher are used to verify the onboarding information, which encompasses the boot image and its hash or signature, initial configuration, and pre- and post-configuration scripts to be executed by the device.

2.9. FIDO Device Onboard (FDO)

The Fast IDentity Online Alliance (FIDO) has specified an automatic onboarding protocol for IoT devices [fidospec]. The goal of this protocol is to provide a new IoT device with information for interacting securely with an online IoT platform. This protocol allows owners to choose the IoT platform for their devices at a late stage in the device lifecycle. The draft specification refers to this feature as "late binding".

The FIDO IoT protocol itself is composed of one Device Initialization (DI) protocol and 3 Transfer of Ownership (TO) protocols TO0, TO1, TO2. Protocol messages are encoded in Concise Binary Object Representation (CBOR) [RFC8949] and can be transported over application layer protocols such as Constrained Application Protocol (CoAP) [RFC7252] or directly over TCP, Bluetooth etc. FIDO IoT however assumes that the device already has IP connectivity to a rendezvous server. Establishing this initial IP connectivity is explicitly stated as "out-of-scope" but the draft specification hints at the usage of Hypertext Transfer Protocol (HTTP) [RFC7230] proxies.

The specification only provides a non-normative example of the DI protocol which must be executed in the factory during device manufacture. This protocol embeds initial ownership and manufacturing credentials into a Restricted Operation Environment (ROE) on the device. The initial information embedded also includes a unique device identifier (called GUID in the specification). After DI is executed, the manufacturer has an ownership voucher which is passed along the supply chain to the device owner.

When a device is unboxed and powered on by the new owner, the device discovers a network-local or an Internet-based rendezvous server. Protocols (TO0, TO1, and TO2) between the device, the rendezvous server, and the new owner (as the owner onboarding service) ensure that the device and the new owner are able to authenticate each other. Thereafter, the new owner establishes cryptographic control of the device and provides it with credentials of the IoT platform which the device should use.

FIDO has the following characteristics:

* *Terms*:

  - _Onboarding_: is the central term used in FDO and represents the complete automated process that transitions a device from its factory state to operational use under the owner's management. It covers the discovery of the owner through the rendezvous server, the execution of TO1 and TO2, and the mutual authentication and secure exchange of configuration and credentials between the device and the owner's onboarding service. The ability to determine the final owner and configuration long after the device has been manufactured is referred to as _late binding_.

  - _Provisioning_: used interchangeably with the term onboaring to refer to the entire process of making the device operational. The introduction section of the specification even states _FDO is a device onboarding scheme from the FIDO Alliance, sometimes called "device provisioning"_

  - _Initialization_: used to refer to the Device Initialization (DI) protocol, which occurs during device manufacturing and installs on the device all initial information, including the attestation key pair, unique identifier (GUID), rendezvous server information, manufacturer public key hash, and the secret used for computing the device HMAC. During the same phase, the manufacturer's tools generate the ownership voucher that corresponds to the device's credentials.

  - _Resale_: refers to the protocol actions that the current device owner can perform to transfer ownership of the device to a new owner without the involvement of the original manufacturer. This is made possible by the ability of the owner to update the device's credentials and generate a new ownership voucher during the onboarding process.

  - _Re-provisioning_: refers to the complete onboarding process being executed again for a new owner.

* *Players*: The device manufacturer plays a significant role in FDO. During production, the manufacturer provisions each device with a unique identifier (GUID), an asymmetric key pair used for cryptographic attestation (based on Intel EPID or standard ECDSA), a hash of the manufacturer's public key (used for verifying the ownership voucher chain), and detailed rendezvous information describing how the device can later locate its onboarding infrastructure. These rendezvous instructions may include DNS names, IP addresses, certificate hashes for TLS pinning, and even information such as a Wi-Fi SSID and passphrase to enable initial connectivity to the rendezvous server. The manufacturer also ensures that the device generates a symmetric secret, which is then used to compute an HMAC value over elements such as the device's GUID, rendezvous information, and manufacturer public key. This HMAC, unique to each unit, is embedded into a signed ownership voucher created by the manufacturer and transferred separately through the supply chain, establishing a verifiable link between the physical device and its digital identity. The specification does not prescribe who operates the rendezvous servers and refers to them generically as Internet services; in practice, they are often maintained by the manufacturer but could be hosted by other entities.

  The final device owner, who seeks long-term management of the device for tasks such as retrieving data and performing software updates, either operates a dedicated onboarding service or delegates this function to another entity. The owner maintains a key pair whose public key is linked to the last entry in the ownership voucher and uses it to register device ownership with a rendezvous server. During the TO2 onboarding protocol, the owner authenticates to the device using its private key and the voucher, while the device attests its identity in return. Once mutual trust is established, the owner's onboarding service securely provisions operational credentials and configuration data, after which the device updates its internal FDO credentials to enable future resale or re-onboarding as needed.

  FDO can be seen as a more elaborate evolution of SZTP. While both rely on manufacturer-installed credentials and trusted servers for redirection, FDO introduces a tighter cryptographic binding between the device and its ownership voucher through the embedded HMAC mechanism and supports verifiable ownership transfers across multiple entities. This design adds complexity but provides the important advantage that devices remain manageable even if the original manufacturer ceases operation, since ownership and onboarding continuity can be maintained by the last legitimate owner, who can update all information except the device's attestation keys.

* *Initial beliefs assumed in the device*: FDO assumes that each device is provisioned during manufacturing with a unique device identifier (GUID), a cryptographic key pair used for device attestation, a hash of the manufacturer's public key for verifying the ownership voucher chain, and the rendezvous information needed to discover onboarding services. The rendezvous information may include DNS names, IP addresses, supported communication protocols, certificate hashes for authenticating the rendezvous server, and even Wi-Fi SSID and passphrase details to enable initial network connectivity. The device also holds a symmetric secret used to compute and verify HMAC values that cryptographically bind the device's internal identity to its ownership voucher.

* *Processes*: A precursor for an FDO device to onboard is the DI protocol, which imparts the information stated above. At the same time, the manufacturer creates an ownership voucher linked to the device through a device-generated HMAC value. The final owner, after receiving this voucher, possibly transferred through several intermediaries in the supply chain, registers the address of its onboarding service with a rendezvous server using the device GUID through the TO0 protocol. When the device is powered on, it follows its stored rendezvous instructions and performs the TO1 protocol to contact a rendezvous server, from which it retrieves the signed blob of data containing the network location of the owner's onboarding service. The device may optionally authenticate the rendezvous server through TLS if the rendezvous information specifies HTTPS and includes certificate hashes for pinning. The device then initiates a direct connection to the onboarding service, where the TO2 protocol is executed to perform mutual authentication and ownership transfer. During this process, the owner proves ownership using the ownership voucher, while the device authenticates itself using its attestation credentials. The device also verifies the authenticity of the signed blob received from the rendezvous server using the now-trusted owner key. After successful verification on both sides, a secure communication channel is established through which the owner's onboarding service transfers operational data such as network configuration parameters, addresses and endpoints of management servers or IoT platforms, authentication credentials such as keys, certificates, tokens, or shared secrets. The device applies these configurations and updates its internal FDO credentials with new information provided by the owner, which may include an updated device identifier, new rendezvous information, and a new trust anchor derived from the owner's public key. Similar to SZTP, once the supporting infrastructure such as the ownership voucher and rendezvous server registration has been set up, FDO enables an experience where only the physical installation

and powering of the device are required for it to securely join the owner's management system or IoT platform. The specification does mention a _Trusted Installer_ mode where the device is provided with advice and input during the onboarding process from the installer, however this is currently not defined and is left for future releases.

* *Beliefs imparted to the device after protocol execution*: After the FDO onboarding process is complete, the device possesses all configuration and credential information necessary for secure operation under the new owner's management domain. This includes data transferred from the owner's onboarding service, which may contain network configuration parameters, the address and endpoint URLs of the final management server or IoT platform, and the credentials such as keys, certificates, tokens, or shared secrets required for the device to authenticate securely to these services. The data may also include instructions for installing required agents, drivers, or firmware updates. In addition, the device may receive and update its internal FDO credentials with new information supplied by the owner, which can include an updated device identifier (GUID), revised rendezvous server information, and a new hash of the owner's public key to establish a renewed trust anchor.

2.10. LPWAN

Low Power Wide Area Network (LPWAN) encompasses a wide variety of technologies whose link-layer characteristics are severely constrained in comparison to other typical IoT link-layer technologies such as Bluetooth or IEEE 802.15.4. While some LPWAN technologies rely on proprietary bootstrapping solutions which are not publicly accessible, others simply ignore the challenge of bootstrapping and key distribution. In this section, we discuss the bootstrapping methods used by LPWAN technologies covered in [RFC8376].

* LoRaWAN [LoRaWAN] describes its own protocol to authenticate nodes before allowing them join a LoRaWAN network. This process is called as joining and it is based on pre-shared keys (called AppKeys in the standard). The joining procedure comprises only one exchange (join-request and join-accept) between the joining node and the network server. There are several adaptations to this joining procedure that allow network servers to delegate authentication and authorization to a backend AAA infrastructure [RFC2904].

* Wi-SUN Alliance Field Area Network (FAN) uses IEEE 802.1X [ieee8021x] and EAP-TLS for network access authentication. It performs a 4-way handshake to establish a session keys after EAP-TLS authentication.

* NB-IoT relies on the traditional 3GPP mutual authentication scheme based on a shared-secret in the Subscriber Identity Module (SIM) of the device and the mobile operator.

* Sigfox security is based on unique device identifiers and cryptographic keys. As stated in [RFC8376], although the algorithms and keying details are not publicly available, there is sufficient information to indicate that bootstrapping in Sigfox is based on pre-established credentials between the device and the Sigfox network.

From the above, it is clear that all LPWAN technologies rely on pre-provisioned credentials for authentication between a new device and the network.

LPWAN has the following characteristics:

* Terms: Provisioning, configuration, discovery.

* Players: Administrator, Authenticator, Border Router, Client, Device, Manager, Network Server, User

* Initial beliefs assumed in the device: The device normally has credentials that are used to directly secure the communications or to device key material to do so. There is a basic trust in the network server.

* Processes: Provisioning

* Beliefs imparted to the device after protocol execution: Either because of an authentication process that results in newly derived key material or the pre-provisioned key material is used, the device is able to exchange information securely through the network server.

2.11. Thread

Thread Group commissioning [threadcommissioning] introduces a two phased process i.e. Petitioning and Joining. Entities involved are leader, joiner, commissioner, joiner router, and border router. Leader is the first device in Thread network that must be commissioned using out-of-band process and is used to inject correct user generated Commissioning Credentials (can be changed later) into

Thread Network. Joiner is the node that intends to get authenticated and authorized on Thread Network. Commissioner is either within the Thread Network (Native) or connected with Thread Network via a WLAN (External).

Under some topologies, Joiner Router and Border Router facilitate the Joiner node to reach Native and External Commissioner, respectively. Petitioning begins before Joining process and is used to grant sole commissioning authority to a Commissioner. After an authorized Commissioner is designated, eligible thread devices can join network. Pair-wise key is shared between Commissioner and Joiner, network parameters (such as network name, security policy, etc.,) are sent out securely (using pair-wise key) by Joiner Router to Joiner for letting Joiner to join the Thread Network. Entities involved in Joining process depends on system topology i.e. location of Commissioner and Joiner. Thread networks only operate using IPv6. Thread devices can devise GUAs (Global Unicast Addresses) [RFC4291]. Provision also exist via Border Router, for Thread device to acquire individual global address by means of DHCPv6 or using SLAAC (Stateless Address Autoconfiguration) address derived with advertised network prefix.

Thread has the following characteristics:

* Terms: Commissioning, discovery, provisioning.

* Players: Administrator, Border Agent, Border Router, Commissioner, Commissioner Candidate, Configurator, Device, End Device, End Device, Endpoint Identifier, Initiator, Joiner, Joiner Router, Owner, Peer, Responder, Server, User

* Initial beliefs assumed in the device: The joiner needs to share credentials with an entity that belongs to the Thread network, prior to the authentication process.

* Processes: Petitioning, Joining

* Beliefs imparted to the device after protocol execution: Once the authentication takes place, a trust relation is established between the Joiner and the Commissioner it receives the network parameters needed to be attached to the Thread network.

3. Comparison

There are several stages before a device becomes fully operational. There is typically a stage where some sort of credential is installed. The nature or purpose of this credential can be varied, form being part of the IoT device authentication, to a credential from a 3rd trusted party, be it the manufacturer or the owner. Solution differ on this initial process, and in some cases this can even be done in an out-of-band fashion.

After some initial credential is installed, there is a process that typically involves authentication establishing initial trust, after which credentials and/or parameters are configured or installed into the device.

Finally, when the entities involved in the process are authenticated and the configuration and key material established, the normal operation of the IoT device can take place.

3.1. Comparison of terminology

The specifics of every term varies depending on the technology, but we enumerate here the basic terminology and what it means for the different solutions.

* Bootstrapping:
  - DPP: Client obtains the Controller’s public bootstrapping key and IP address
  - OMA: An IoT device retrieves and processes all the bootstrap data
  - EST: installation of the Explicit TA database
  - BRSKI: A protocol to obtain a local trust anchor.
  - SZTP: The process by which obtains "bootstrapping data" such as conveyed information, owner certificate and owner voucher.
  - EAP-NOOB: For an IoT device to be registered, authenticated, authorized and for it to derive key material to act as a trustworty entity in the security domain where it is deployed.
* configuration:
  - DPP: The process performed by a Configurator by which the Enrollee is provisioned.

- OMA: Adding or removing an LwM2M Server Account to or from the LwM2M Client configuration.

- OCF: The necessary information the Device must hold to be considered as ready for normal operation.

- EST: The basic information (e.g., TA database) needed to initiate protocol operation.

- SZTP: The system configuration to be installed into the device by the bootstrapping process.

- EAP-NOOB: Establishing necessary information for the device to operate.

- LPWAN: In LoRaWAN, the information related to the working of the device and protocol.

* discovery:

  - DPP: Exchange that allows obtaining specific information such as SSID, operating channel and band.

  - OCF: Making the different resources available through URIs.

  - BRSKI: Locating an entity that needs to take part of the bootstrapping process (e.g., Join proxy)

* enrollment:

  - EST: The process of obtaining the credentials needed to perform the device normal operation.

  - BRSKI: Same process describe as EST.

  - SZTP: The process of an owner joining a manufacturer’s SZTP program.

* provisioning:

  - DPP: Securely enabling a device to establish secure associations with other devices in a network.

  - OMA: Establishing security credentials and access control lists by a dedicated LwM2M bootstrap server.

- OCF: A set of processes that take place both during and after the ownership transfer. These entail configuration of credentials, and security-related resources for any services or devices that the provisioned device needs to interact with in the future.

- Bluetooth: The procedure by which a device is authenticated, and a secure link is established, becoming a trustworthy node in the network.

- FIDO: Same as FIDO onboarding.

- SZTP: The set of steps that take place to enable a device to establish secure connections with other systems.

- LPWAN: In LoRaWAN, the establishment of configuration data and credentials.

* intialization:

  - OMA: When Bootstrap-Delete operation is used, to restore a device.

  - FIDO: Protocol (DI), establishing basic information at manufacture.

* registration:

  - OMA: Establishing a registration session, which is an association between the client and the server.

  - EAP-NOOB: Add information about an IoT device in a server database.

* onboarding:

  - OCF: The device is considered to complete the onboarding after the ownership of the Device has been transferred and the Device provisioned.

  - FIDO: The procedure of installing configuration information and secret to a device so that it may safely connect to and communicate with an IoT platform.

  - SZTP: information related to the boot image a device must be running, an initial configuration the device must commit, and scripts that the device must successfully execute.

* commissioning:

  - Thread: The process of a Thread device joining a Thread network.

* imprint:

  - BRSKI: The process by which a device obtains the needed information to act as trustworthy entity within the network or domain

3.2. Comparison of players

In this section we classify the different players.

Human User: user

Device that intends to securely join a network: pledge, device, client, peer, persona, enrollee, candidate

Entity that makes the device: Manufacturer

Entity that owns the device: owner, manager

Entity with which the device establishes a connection: IoT platform, Rendezvous Server, Server,

Entity that aids in the process: Join Proxy, Bootstrap Server, Onboarding Server, Border Router

Person that manages a deployment or system: Administrator

Entity that steers the process for the IoT device to securely join the network: Configurator, Bootstrap Server, Rendezvous Server, JRC, Onboarding/Redirect Server, Commissioner.

External or third-party entity that intervenes in the process: Registrar, MASA

3.3. Comparison of initial beliefs

The IoT devices may have different initial beliefs depending on the credentials pre-installed, during the manufacturing process or prior to being turned on. There are cases where the initial credentials that need to be shared to establish basic trust, or they are exchanged during one of the procedures after the device is turned on, not installed during manufacturing.

3.3.1. No initial trust established

EAP-NOOB does not require initial configuration of credentials to establish trust, since its done using the out-of-band process.

The OCF device starts as unowned. It has to perform an ownership transfer, to establish basic trust to perform onboarding and provisioning. Depending on the Owener Transfer Mode (OTM) it can be considered to have not initial trust based on the credentials installed.

Bluetooth devices start as unprovisioned. Initial trust is established as a consequence of exchanging public keys and performing the authentication. If the public keys are ephemeral, there is no initial credential establishment.

3.3.2. Initial trust based on the credentials installed

These credentials may very from the time of installing, and the entity to which it related. In this sense, they could be from the manufacturer, owner or other entity.

FIDO devices have installed during the manufacturing process a set of ownership credentials (i.e., ownership voucher) and additional information to determine the current owner of the device. Hence, there is an initial trust from the IoT device and the owner. With this basic setup, and and the cooperation among device, owner and rendezvous server, the onboarding process can take place.

EST devices are configured with the needed information to perform mutual authentication and for authorization between the EST client and server.

BRSKI have manufacturer-installed certificates as starting point to establish trust.

SZTP have pre-configured initial state which provides the basis for trust.

LPWAN specifics depends on the technology, but they all have in common some pre-installed credentials that allows the establishment of trust and to secure the communications.

Thread devices, share credentials as well to establish trust.

OMA devices can have all the necessary information to start working on the network where they are deployed, if they have the factory bootstrap, hence all needed credentials to establish trust. There need to be installed some basic credentials to establish trust with the bootstrap server and perform the bootstrapping.

DPP initial trust is established during the bootstrapping where the public key is transmitted.

3.4. Comparison of processes

Analyzing the different terms used over the different solutions reviewed in this document, we can identify several processes. These are named differently in some cases, and not every technologies considers them all as part of their the following common concepts:

* To refer to the process previous to the device being turned on, in which some information, or credentials are installed into the device. This process is commonly referred to as manufacture. Is in this phase where the IoT devices have installed the needed information (specifics depend on the technology) to provide the basis for trust and to authenticate other entities.

* To refer to the process after the device is turned on and intends to locate the entity with which it has to communicate to start the authentication process to be integrated into the security domain. Here is where the device start the process to get to perform its normal operation.

* To the process by which the device obtains additional credentials, in addition to what it already had installed before being turned on.

* To the process by which the device is authenticated and established a trust relation.

3.5. Comparison of knowledge imparted to the device

Even though the devices might start from a different place, in terms of initial credentials as basis for trust, when they finish their processes, they become trusted parties within the domain they are deployed or at least have a trust relation with a specific entity. The difference may strive in the number of trust relations are stablished during the process, as they may have not only established a trust relation with local entities where they perform their operation, but other external entities as well.

In FIDO, once the onboarding process has taken place, the IoT device is mutually authenticated with the current owner, and the needed secrets and configuration data is installed into the device, which as a result is able to connect and interact securely with the target IoT platform.

In EAP-NOOB, once the bootstrapping is completed, the IoT device not only has a trust relation with the EAP server, but the EAP authenticator can be established as well, based on the shared credentials that are derived during the authentication process.

In Bluetooth the trust is expanded to the local network as there is key material shared among the different entities of the network.

In Thread once the joiner has successfully completed the process, it can communicate directly with all Thread devices in the network.

LPWAN has a more limited scope and they usually have specific keys for applications and network communications.

EST provides the devices with the enrollment information such as certificates and symmetric keys that can be used to establish trust with different peers.

BRSKI after running it is able to verify that the communicating entities are who they claim to be, and obtain domain specific certificates to act as trustworhty entities within the domain.

SZTP after running, the device has obtained onboarding information and is equipped to establish secure connections with other systems.

4. Security Considerations

This draft does not take any posture on the security properties of the different bootstrapping protocols discussed. Specific security considerations of bootstrapping protocols are present in the respective specifications.

Nonetheless, we briefly discuss some important security aspects which are not fully explored in various specifications.

Firstly, an IoT system may deal with authorization for resources and services separately from initial security setup in terms of timing as well as protocols. As an example, two resource-constrained devices A and B may perform mutual authentication using credentials provided by an offline third-party X before device A obtains authorization for running a particular application on device B from an online third-party Y. In some cases, authentication and authorization maybe tightly coupled, e.g., successful authentication also means successful authorization.

Secondly, initial security setup of IoT devices may be necessary several times during the device lifecycle since keys have limited lifetimes and devices may be lost or resold. Protocols and systems must have adequate provisions for revocation and fresh security setup. A rerun of the security setup mechanism must be as secure as the initial security setup regardless of whether it is done manually or automatically over the network.

Lastly, some IoT networks use a common group key for multicast and broadcast traffic. As the number of devices in a network increase over time, a common group key may not be scalable and the same network may need to be split into separate groups with different keys. Bootstrapping and provisioning protocols may need appropriate mechanisms for identifying and distributing keys to the current member devices of each group.

5. IANA Considerations

There are no IANA considerations for this document.

6. Acknowledgements

We would like to thank Tuomas Aura, Hannes Tschofenig, and Michael Richardson for providing extensive feedback as well as Rafa Marin-Lopez for his support.

7. Informative References

[dpp] Wi-Fi Alliance, "Wi-Fi Easy Connect Specification", Wi-Fi Alliance Version 3.0, 2018, <https://www.wi-fi.org/wi-fi-download/35330>.

[fidospec] Fast Identity Online Alliance, "FIDO Device Onboard Specification", Fido Alliance Version: 1.1, April 2022, <https://fidoalliance.org/specifications/download-iot-specifications/>.

[ieee8021ar]
IEEE, "IEEE Standard for Local and metropolitan area networks寰鉄ecure Device Identity", 2018, <https://1.ieee802.org/security/802-1ar/>.

[ieee8021x]
IEEE, "IEEE Standard for Local and metropolitan area networks寰撤ort-Based Network Access Control", 2020, <https://1.ieee802.org/security/802-1x/>.

[LoRaWAN] LoRa Alliance, "LoRa Specification", LoRa Alliance Version: 1.1, October 2017, <https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/>.

[ocf] Open Connectivity Foundation, "OCF Security Specification", Version 2.2.6, October 2022, <https://openconnectivity.org/specs/OCF_Security_Specification.pdf>.

[oma] Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification: Core", Approved Version 1.2.1, December 2022, <https://openmobilealliance.org/release/LightweightM2M/V1_2_1-20221209-A/OMA-TS-LightweightM2M_Core-V1_2_1-20221209-A.pdf>.

[oma-transport]
Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification: Transport Bindings", Approved Version 1.2.1, December 2022, <https://www.openmobilealliance.org/release/LightweightM2M/V1_2_1-20221209-A/OMA-TS-LightweightM2M_Transport-V1_2_1-20221209-A.pdf>.

[RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, DOI 10.17487/RFC2904, August 2000, <https://www.rfc-editor.org/rfc/rfc2904>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/rfc/rfc4291>.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <https://www.rfc-editor.org/rfc/rfc5272>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/rfc6241>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/rfc/rfc7030>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/rfc/rfc7228>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/rfc/rfc7230>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/rfc8040>.

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/rfc/rfc8366>.

[RFC8376] Farrell, S., Ed., "Low-Power Wide Area Network (LPWAN) Overview", RFC 8376, DOI 10.17487/RFC8376, May 2018, <https://www.rfc-editor.org/rfc/rfc8376>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/rfc/rfc8572>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/rfc/rfc8995>.

[RFC9140] Aura, T., Sethi, M., and A. Peltonen, "Nimble Out-of-Band Authentication for EAP (EAP-NOOB)", RFC 9140, DOI 10.17487/RFC9140, December 2021, <https://www.rfc-editor.org/rfc/rfc9140>.

[RFC9148] van der Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol", RFC 9148, DOI 10.17487/RFC9148, April 2022, <https://www.rfc-editor.org/rfc/rfc9148>.

[Sethi14] Sethi, M., Oat, E., Di Francesco, M., and T. Aura, "Secure Bootstrapping of Cloud-Managed Ubiquitous Displays", Proceedings of ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2014), pp. 739-750, Seattle, USA, September 2014, <http://dx.doi.org/10.1145/2632048.2632049>.

[simpleconn] Wi-Fi Alliance, "Wi-Fi Simple Configuration", Version 2.0.7, 2019, <https://www.wi-fi.org/download.php?file=/sites/default/files/private/Wi-Fi_Simple_Configuration_Technical_Specification_v2.0.7.pdf>.

[threadcommissioning] Thread Group, "Thread Commissioning", 2015.

Authors' Addresses

Mohit Sethi
Aalto University
FI-02150 Espoo
Finland
Email: mohit@iki.fi

Behcet Sarikaya
Email: sarikaya@ieee.org

Dan Garcia-Carrillo
University of Oviedo
33207 Oviedo
Spain
Email: garciadan@uniovi.es