Network Working Group
Internet-Draft
Intended status: Informational
Expires: 21 July 2026

M. Sethi
Aalto University
B. Sarikaya
Unaffiliated
D. Garcia-Carrillo
University of Oviedo
17 January 2026

Terminology and processes for initial security setup of IoT devices
draft-irtf-t2trg-security-setup-iot-devices-06

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 21 July 2026.

Copyright Notice

Table of Contents

1. Introduction 2
2. Standards and Protocols 4
2.1. Bluetooth Mesh Provisioning 4
2.2. Device Provisioning Protocol (DPP) 9
2.3. Enrollment over Secure Transport (EST) 12
2.4. Open Mobile Alliance (OMA) Lightweight Machine to Machine specification (LwM2M) 14
2.5. Nimble out-of-band authentication for EAP (EAP-NOOB) . . 17
2.6. Open Connectivity Foundation (OCF) 18
2.7. Bootstrapping Remote Secure Key Infrastructures (BRSKI) 22
2.8. Secure Zero Touch Provisioning (SZTP) 25
2.9. FIDO Device Onboard (FDO) 28
2.10. Thread 32
3. Comparison 35
3.1. Comparison of terminology 36
3.2. Comparison of players 38
3.3. Comparison of initial beliefs 39
3.4. Comparison of processes 42
3.5. Comparison of knowledge imparted to the device 43
3.6. Other observations 45
4. Security Considerations 45
5. IANA Considerations 45
6. Acknowledgements 46
7. Informative References 46
Authors' Addresses 49

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 Mesh Provisioning

Bluetooth Mesh [bt-mesh] specifies a provisioning protocol to securely incorporate a new device into an existing Bluetooth mesh network. Provisioning includes authenticating an unprovisioned device and securely delivering network specific secrets and addressing information that allow the device to become a mesh node. The process relies on a Provisioner, typically a smartphone application or a dedicated management device, which interacts with the unprovisioned device (called provisionee) and assigns it a unicast address and one or more Network Keys. Once provisioned, the device becomes a _node_ in the mesh network. Within the network, one or more nodes may take on the role of Configuration Manager, using device specific Device Keys to configure nodes and distribute Application Keys that are required for secure application layer communication.

Bluetooth Mesh provisioning proceeds through several well defined phases. It begins with beaconing and discovery, where an unprovisioned device advertises its presence and Device UUID, allowing a Provisioner to identify it. This is followed by a capabilities exchange, during which the device reports its supported cryptographic algorithms, available authentication methods, and I/O

capabilities. The Provisioner then selects an authentication method and the two parties perform an ECDH key exchange. The provisionee public key may be static and transferred using an out of band mechanism, or ephemeral and sent directly over-the-air bearer. Both sides derive a shared ECDH secret, which is used to establish session keys that protect the remainder of the provisioning protocol. During the authentication phase, the method selected based on the capabilities exchange, namely Output OOB, Input OOB, Static OOB, or No OOB, is used to authenticate the key exchange and confirm user intent. The selected input or output method determines the direction in which a random value, either encoded into the device during manufacturing or generated at runtime, is transferred. Once authentication is complete, the Provisioner securely distributes the provisioning data, including the Network Key, unicast address, and other network parameters. After this step, the device is fully provisioned and can participate securely in mesh communication.

Since Bluetooth Mesh version 1.1, certificate based provisioning is also supported. In this mode, the unprovisioned device is provisioned during manufacturing with a device certificate that binds its public key and UUID to a certificate authority operated by the vendor. During provisioning, the device sends this certificate to the Provisioner, which verifies it against a configured trust anchor certificate authority. In this case, the user can view a list of unprovisioned devices and select the device intended for provisioning without performing explicit actions to transfer keys or random values using out of band mechanisms. The user may be assisted by the selected device providing a visual or audible indication to confirm physical identity. Alternatively, the user may scan a QR code or device URI printed on the device or its packaging, which encodes the UUID. The Provisioner then receives the corresponding certificate containing the same UUID and provisions the device. This enables automated authentication of device identity without relying solely on user mediated out of band mechanisms. Bluetooth Mesh also supports remote provisioning, where the Provisioner does not need to be in direct radio range of the unprovisioned device. Instead, provisioning messages are relayed through already provisioned mesh nodes, allowing devices to be added to large or geographically distributed deployments without local access.

Bluetooth mesh has the following characteristics:

* *Terms*:

- _Provisioning_: is the central term used to refer to the complete process of securely adding a new device, called a provisionee, to an existing mesh network with the help of a Provisioner. It includes discovery, authentication, establishment of shared keys, and delivery of network specific credentials and addressing information.

- _Beaconing and discovery_: describe the mechanism by which unprovisioned devices announce their presence to Provisioners so that the provisioning process can be initiated.

- _Configuration_: describes the post provisioning process by which a node is prepared for participation in the mesh network. Configuration includes the distribution of Application Keys for instance. Note, distribution of the Network Key, unicast address, and Device Key is considered essential and is therefore part of provisioning rather than the optional configuration phase.

- _Reprovisioning_: refers to the process of provisioning a device again, either to add it to a new mesh network or to restore it after removal from a previous network.

* *Players*: In deployments where the device being provisioned uses a static asymmetric key pair, the device manufacturer is responsible for equipping each device with that key pair and making the corresponding public key and related metadata available to the user, for example by printing it on the device or its packaging, often encoded as a QR code or URI. In certificate based provisioning deployments, the manufacturer additionally provisions the device with a unique X.509v3 device certificate. The Provisioner plays a central role in the protocol and is typically implemented as a smartphone application provided by the device manufacturer or system integrator. The device owner/user is responsible for initiating the provisioning process and selecting the correct physical device to add to the mesh network. Depending on the provisioning method, the user may scan a QR code or URI containing the device public key or UUID. If Output OOB, Input OOB, or Static OOB authentication is negotiated, the user is also responsible for transferring the required out of band information, for example by observing a blinking LED or numeric display on the device and entering the value into the Provisioner, or vice versa. When using certificate based provisioning, the user may scan the UUID printed on the device or its packaging, or select the device from a list of unprovisioned devices identified by vendor, model, and UUID. The user may be assisted in selecting the correct physical device being provisioned through visual or audio cues emitted by the device. In larger deployments, already

provisioned mesh nodes may relay provisioning messages, enabling remote provisioning of devices that are not within direct radio range of the Provisioner.

* *Initial beliefs assumed in the device*: Each device must have a unique Device UUID, which is used during discovery to identify the device to a Provisioner. The device may also be provisioned with a device specific static asymmetric key pair that is used in the ECDH key exchange during provisioning. In this case, the public key is typically made available to the Provisioner out-of-band via a QR code or NFC tag printed on the device or its packaging. In deployments that use certificate based provisioning, the device is additionally provisioned at manufacturing time with an X.509 Device Certificate that binds the device public key and the UUID. The corresponding trust anchor CA for validating this certificate is assumed to be available to the Provisioner.

* *Processes*: Once an unprovisioned device is powered on, it advertises its presence using unprovisioned beacons. The user or owner then uses a Provisioner application to discover nearby unprovisioned devices. The user may select the correct device based on information such as make, model, and Device UUID displayed by the Provisioner. In deployments that use static public keys or certificate based provisioning, the user may also scan a QR code or URI on the device or its packaging to transfer the device public key and or UUID to the Provisioner prior to authentication. Thereafter, the unprovisioned device and the Provisioner perform a capabilities exchange to determine supported cryptographic algorithms, authentication methods, and available input and output interfaces. At this stage, the user or owner may be assisted by visual or audio cues emitted by the device, as permitted by its reported capabilities, to clearly identify the physical device that is being provisioned. Based on the exchanged capabilities, the Provisioner selects an authentication method and initiates an Elliptic Curve Diffie Hellman key exchange. The key exchange is then authenticated using one of the supported methods, such as Input OOB, Output OOB, or Static OOB, in which case the user assists by transferring or confirming authentication data between the device and the Provisioner. The specification also allows certificate based authentication, where the device presents a manufacturer issued certificate that is verified by the Provisioner, removing the need for user mediated out of band transfer. Upon successful authentication, the Provisioner derives session keys and a Device Key and securely delivers the provisioning data, including the Network Key and unicast address, to the device. After provisioning is complete, the device becomes a node in the mesh network and can securely participate in network communication. Subsequent configuration is performed using the Device Key to distribute Application Keys.

* *Knowledge imparted to the device after protocol execution*: After the provisioning process is complete, the device possesses one or more Network Keys that enable it to encrypt and authenticate mesh network messages. The device is also assigned a unique unicast address that defines its identity and addressing within the mesh network. In addition, the device holds a Device Key that is shared only with the Provisioner or Configuration Manager and is used to authenticate and protect future configuration messages without requiring reprovisioning. The device may also receive one or more Application Keys that allow it to participate in specific application level communication contexts.

2.2. Device Provisioning Protocol (DPP)

The Wi-Fi Alliance Device Provisioning Protocol (DPP) [dpp] (also called Wi-Fi Easy Connect) is a standardized protocol for providing user-friendly Wi-Fi setup for devices. DPP relies on a configurator, e.g. a smartphone application, for setting up all other devices, called enrollees, in the local network. DPP also supports cloud managed environments where the configurator is not directly reachable on the local network. In this variant, referred to as DPP over TCP, the enrollee communicates with a local relay, typically the Wi Fi access point (AP), which encapsulates DPP messages into a TCP connection to a remote configurator.

In DPP, an enrollee is typically provisioned during manufacturing with a static bootstrapping asymmetric key pair. The configurator may also possess a static bootstrapping key pair generated when the application is first installed or initialized, or it may generate a bootstrapping key pair dynamically. DPP defines four conceptual phases. During the initial _bootstrapping_ phase, the most common deployment model uses unidirectional authentication of the enrollee bootstrapping public key. In this case, the configurator possesses an authenticated copy of the enrollee public key, for example obtained through a QR code printed on the device packaging, while the enrollee does not authenticate the configurator at this stage and instead relies on the configurator having prior knowledge of its public key. Bidirectional authentication of bootstrapping public keys is also supported when enrollee and configurator input and output capabilities permit it, such as through PKEX, where the user inputs a short shared secret code on both sides, or through bidirectional NFC exchanges.

After successful unidirectional or bidirectional bootstrapping authentication, the protocol proceeds to the _authentication_ phase, during which the configurator and enrollee perform a mutually authenticated key exchange using fresh ephemeral keys and the previously authenticated bootstrapping keys. At the conclusion of this phase, both parties derive shared key material. This key material is then used in the _configuration_ phase to securely provision the enrollee with network access information. This includes a Connector (a modern credential containing a tuple of the network identity and an access key instead of a network wide password) and traditional network parameters such as the SSID. In the final _access_ phase, the enrollee uses the Connector together with the network introduction protocol to authenticate to and establish connectivity with the Wi Fi access point (AP), completing the DPP provisioning process.

DPP has the following characteristics:

* *Terms*:

  - _Bootstrapping_: refers to the initial establishment of trust between the configurator and the enrollee through an out of band exchange of bootstrapping public keys. Only concerned with public key authentication rather than network configuration.

  - _Discovery_: refers to the mechanisms by which devices locate each other and determine how DPP messages should be exchanged.

  - _Enrollment_: used informally to refer to the act of authorizing a device to join a specific network domain

  - _Configuration_: refers to the phase in which the configurator provisions the enrollee with the information required to join a network domain. This includes delivery of a DPP configuration object containing network parameters and a Connector.

  - _Reconfiguration_: refers to the ability of an already provisioned device to obtain updated network configuration information from a configurator without repeating the initial bootstrapping phase.

  - _Provisioning_: used more broadly in DPP to describe the overall process of securely configuring a device for network access. It encompasses bootstrapping, authentication, configuration, and access, rather than referring to a single protocol step.

* *Players*: The device manufacturer is responsible for provisioning each device with a bootstrapping asymmetric key pair during manufacturing. In many deployments, the manufacturer is also responsible for encoding the corresponding bootstrapping public key, together with associated metadata, as a QR code printed on the device or its packaging. The device owner/user is responsible for securely transferring this bootstrapping public key information to the configurator. The configurator required to perform the DPP protocol and enable new devices to join the Wi Fi network can be provided through functionality built into mobile operating systems such as Android or iOS, or through a dedicated application supplied by the device manufacturer. In enterprise or cloud managed deployments, the configurator may be part of a remote IoT platform, in which case the local Wi-Fi access point acts as a relay and must be configured with the network location of the remote configurator.

* *Initial beliefs assumed in the device*: DPP requires devices to have a bootstrapping asymmetric key pair, which is typically installed in the factory. The corresponding bootstrapping public key is intended to be conveyed securely to the configurator through an out of band mechanism. Depending on the selected DPP variant and the device capabilities, this may require the device to support specific I/O interfaces. For example, devices may present the bootstrapping public key and associated metadata encoded as a QR code or NFC tag, or they may support limited user input to enable password based bootstrapping using PKEX.

* *Processes*: DPP provisioning process begins with bootstrapping, during which a user initiates an out-of-band transfer of public keys between the configurator and the enrollee. In a typical unidirectional deployment, the user employs a configurator, such as a smartphone application, to scan a QR code associated with the enrollee, thereby providing the configurator with an authenticated copy of the enrollee bootstrapping public key. Bidirectional variants are also supported, including NFC based exchanges and PKEX, where both the configurator and the enrollee authenticate each other using a short-shared secret input by the user. Once the configurator possesses the enrollee authenticated bootstrapping key, the devices proceed to the authentication phase, performing cryptographic exchanges to verify the bootstrapping keys and establish a secure encrypted channel using derived shared key material. During the subsequent configuration phase, the configurator uses this protected channel to provision the enrollee with a DPP configuration object containing deployment specific network parameters and a Connector. The Connector is a cryptographically protected credential that binds the network identity to a device specific access key and replaces legacy network wide passwords. In the access phase, the enrollee presents the Connector to a Wi-Fi AP to establish secure association with the network. If network settings change or credentials expire, the device can also undergo reconfiguration by using its stored keys to obtain updated configuration information without repeating the initial physical bootstrapping steps.

* *Knowledge imparted to the device after protocol execution*: After successful completion of the DPP provisioning process, the enrollee device and the configurator establish shared key material that is used to protect the subsequent information provisioned onto the device. The information includes a Connector, which is a signed and cryptographically protected structure containing a group identifier, a network role, and a network access key. The Connector replaces traditional network wide passwords and allows the device to prove to other peers that it has been authorized by the configurator to join the network. The device also receives

and trusts the configurator public signing key, which it uses to verify Connectors presented by other devices such as APs. In addition, the device gains knowledge of deployment specific network parameters, typically including the SSID, supported operating channels, security policies of the domain, and privacy protection keys (used to protect identity of the enrollee when requesting reconfiguration by the configurator).

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 [RFC7617] or Digest [RFC7616] 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.

* *Knowledge 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.

* *Knowledge 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. The devices also populate the PeerInfo JSON object using during the boostrapping process and may contain information such as the MAC Address, Manufacturer name, and model information.

* *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.

* *Knowledge 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 set of procedures required to bring a device from an unowned, factory default state into an operational and managed state as onboarding. A central element of onboarding is ownership transfer, which establishes a legitimate device owner and forms the root of trust for all subsequent security configuration and access control. Ownership transfer is performed with the assistance of an Onboarding Tool (OBT), a logical entity under the control of the user or administrator that interacts with the device to authenticate it, establish secure communication, and provision initial security credentials. 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 specifies several standardized Ownership Transfer Methods (OTMs) that define how the device and the OBT authenticate each other and establish a secure channel during onboarding. These include Just Works, which performs an unauthenticated Diffie寰滴ellman exchange using DTLS and provides confidentiality but no protection against on-path attackers; Random PIN, where the device presents a user-visible PIN that is entered into the OBT and used as the PSK for DTLS-PSK to authenticate the exchange; and Manufacturer Certificate-based ownership transfer, in which the device authenticates itself using a manufacturer-installed X.509 certificate and the OBT verifies this certificate against a configured CA trust anchor. In certificate-based OTMs, mutual authentication may also be achieved if the device validates the OBT寰洉 credentials. OCF additionally allows vendor-specific OTMs to support proprietary hardware capabilities or deployment-specific requirements, provided they conform to the overall security architecture.

Upon successful completion of an OTM, the OBT provisions the device with Owner Credentials, formally transitioning the device from an unowned to an owned state. These credentials establish a long-term trust relationship between the device and its owner and may take the form of certificates, shared secrets, or other credential types defined by the OCF security framework. As part of onboarding, the OBT also provisions information required for interaction with core OCF security services, including the Access Management Service (AMS) and the Credential Management Service (CMS). The AMS is responsible for managing access control policies and permissions, while the CMS manages the lifecycle of security credentials used by the device during normal operation. Together, these steps ensure that the device is securely owned, authenticated, and ready for controlled participation in an OCF ecosystem.

OCF has the following characteristics:

* *Terms*:

  - _Onboarding_: is the umbrella process that transitions a device from an initial unowned state to a fully operational managed state and includes discovery, ownership transfer, credential provisioning, and initialization of access control and credential management services.

  - _Discovery_: process by which an Onboarding Tool (OBT) locates unowned or owned devices on the network and learns their basic identifiers, security state, and supported Ownership Transfer Methods.

- _Enrollment_: used in the context of certificate enrollment to obtain a certificate from the CMS after onboarding is complete.

- _Configuration_: refers to the act of setting or modifying the state of device resources through their exposed interfaces. Configuration applies to both functional resources and security related resources such as identity and credentials.

- _Provisioning_: refers to security focused configuration operations performed during and after ownership transfer that establish or modify the trust relationships of a device. Provisioning includes the configuration of ownership information, security credentials, trust anchors, access control policies, and security service endpoints, and defines the security domain in which the device operates.

- _Registration_: act of associating a device with its security domain and management services after ownership transfer. Registration typically includes provisioning the locations and identities of the Access Management Service (AMS) and Credential Management Service (CMS).

* Players: The device manufacturer is responsible for provisioning the device during manufacturing with a unique device identity (UUID) and, where certificate based ownership transfer is used, a manufacturer issued X.509v3 certificate and corresponding trust anchors. The Onboarding Tool (OBT) may be implemented as a smartphone application, a home gateway, or a dedicated device management system, and is often provided by the device manufacturer or platform vendor. The device owner/user is responsible for initiating the onboarding process and establishing ownership of the device using the OBT. During the ownership transfer process, the user may be required to input a randomly generated PIN or explicitly approve a Just Works transfer, with the expectation that this occurs in a physically and logically secure environment. For certificate based ownership transfer methods, the user or owner typically selects the correct device from a list presented by the OBT, identified by its Device UUID or additional device metadata. After onboarding is complete and information about the Credential Management Service (CMS) and the Access Management Service (AMS) has been provisioned into the device, these services become responsible for ongoing security provisioning, including credential lifecycle management, access control configuration, and authorization policy updates.

* *Initial beliefs assumed in the device*: OCF requires devices to have a unique device identityty (UUID). Depending on the supported Ownership Transfer Methods, the device may also have a

manufacturer issued X.509 certificate and corresponding private key. Additionally, the device is initialized with a default, restrictive Access Control List (ACL) that permits anonymous, unauthenticated access only to the specific security resources required for onboarding while blocking access to all other device resources until ownership is established.

* Processes: OCF onboarding process begins when a device that is in an unowned or reset state is discovered by an Onboarding Tool (OBT) using OCF discovery mechanisms. Once discovered, the OBT initiates an Ownership Transfer Method (OTM) to establish a secure relationship between the device and the new owner. Depending on the selected OTM, the device and OBT perform an authenticated or unauthenticated key establishment procedure, such as a Just Works Diffie Hellman exchange, a PIN based authenticated key exchange, or a certificate based authentication using manufacturer installed credentials. During this phase, a secure communication channel is established and ownership of the device is transferred by provisioning an Owner Credential and updating the device ownership state. Upon successful ownership transfer, the device accepts privileged configuration operations from the OBT, regardless of existing access control entries. The OBT then provisions the device with security relevant information required for operational use, including credentials and trust anchors for interacting with the Credential Management Service (CMS) and the Access Management Service (AMS), as well as initial access control policies. After these provisioning steps are completed, the device transitions into an owned and operational state. Subsequent security provisioning and configuration operations are performed through the CMS and AMS, allowing credentials, access control entries, and other security parameters to be updated over the device lifetime without repeating the ownership transfer process.

* *Knowledge imparted to the device after protocol execution*: Upon successful completion of the OCF onboarding and Ownership Transfer process, the device transitions from an unowned state to an owned and operational state. The device now possesses an Owner Credential, which cryptographically binds the device to its legitimate owner and allows it to mutually authenticate the Device Ownership Transfer Service (DOTS) in the future. This credential may take the form of a shared secret, a public key certificate, or another credential type defined by the selected Ownership Transfer Method. In addition, the device is provisioned with the identities and network locations of core OCF security services, including the Credential Management Service (CMS) and the Access Management Service (AMS). The device trusts these services to manage security credentials and access control policies on behalf of the owner.

2.7. Bootstrapping Remote Secure Key Infrastructures (BRSKI)

Bootstrapping Remote Secure Key Infrastructures (BRSKI) [RFC8995] defines a bootstrapping solution that enables devices to securely join the device owner's network domain using manufacturer-installed IEEE 802.1AR [ieee8021ar] certificates, together with a manufacturer-provided Internet service called the Manufacturer Authorized Signing Authority (MASA).The document highlights that the solution is aimed in general at non-constrained (i.e. class 2+ defined in [RFC7228]) devices operating in a non-challenged network. The goal of the protocol is to securely provide a new device (called pledge in the specification) with the CA fingerprint of the owner's network domain, allowing the device to identify and trust future interactions within the owner network. If the owner network provides an Enrollment over Secure Transport (EST) service, the device may also enroll and obtain a locally issued certificate bound to the owner domain. To enable this process, the owner operates a registrar that authenticates devices using their manufacturer installed certificates and requests authorization vouchers from the MASA based on its list of trusted manufacturers and device serial numbers. Prior to full network access, the device may rely on link local connectivity via an owner provided proxy (Join Proxy), allowing it to complete authentication and trust establishment before being assigned a routable network address.

BRSKI has the following characteristics:

* *Terms*:

  - _Bootstrapping_: used to describe the overall process by which a device transitions from an unaffiliated factory state to being a trusted member of an owner network. In BRSKI, bootstrapping encompasses device discovery of the local domain, interaction with the registrar and manufacturer authorized signing authority, validation of a voucher, and establishment of trust in the owner domain. The term broadly covers all protocol steps required before the device can securely operate in the network.

  - _Provisioning_: sed in a general sense to refer to the act of supplying a device with the information and trust anchors it needs to operate securely in a specific domain. In BRSKI, provisioning primarily occurs when the device accepts the domain trust anchor conveyed in the voucher and optionally receives a locally issued device certificate. The term is not used as a formally distinct protocol phase and is largely interchangeable with bootstrapping in descriptive text.

- _Enrollment_: used to refer specifically to the process of obtaining a locally issued identity within the owner domain. This typically occurs after imprinting and is realized through Enrollment over Secure Transport, where the device enrolls with a local certification authority to obtain a Locally Issued Device Identifier certificate.

- _Onboarding_: not used as a primary or formally defined term. It appears informally to generally refer to the broader act of bringing a device under the control of an owner network.

- _Join_: used to describe the act of a device attempting to attach itself to a specific owner network. It reflects the device intent to become a member of the domain but does not by itself imply trust has been established.

- _Imprint_: used to describe the transition where the device accepts the owner domain trust anchor provided in the voucher. Imprinting marks the point at which the device irrevocably binds itself to the owner domain by trusting the pinned domain certificate.

* *Players*: The device manufacturer is responsible for installing the IEEE 802.1AR IDevID certificate and private key during manufacturing, along with the trust anchor required to verify vouchers signed by the Manufacturer Authorized Signing Authority (MASA) and the MASA service location itself. The manufacturer must also operate the MASA service, which authorizes devices to join specific owner domains by issuing signed vouchers, and maintain sufficient asset tracking to associate device identities such as serial numbers with authorized owners.

  On the owner side, the network into which the device is deployed must provide initial connectivity, often via a proxy, to allow the device to communicate with the registrar and indirectly with the MASA before it has full network access. The device owner operates a registrar that authenticates devices using their manufacturer installed certificates, maintains a list of trusted manufacturers and corresponding MASA services, and tracks the identities of devices that are permitted to join the domain. If the owner intends to issue locally scoped identity credentials, the owner must additionally deploy and operate an Enrollment over Secure Transport (EST) server to provision domain specific certificates to devices after successful bootstrapping.

* *Initial beliefs assumed in the device*: BRSKI requires each device to possess an IEEE 802.1AR Initial Device Identifier (IDevID) certificate and associated private key. In addition, the

device must include a built in trust anchor that enables it to validate vouchers signed by the manufacturer寰洯 Manufacturer Authorized Signing Authority (MASA). The device is also configured with the MASA service location, typically expressed as a URI embedded in the IDevID certificate.

* *Processes*: BRSKI assumes that devices are manufactured with an IEEE 802.1AR Initial Device Identifier certificate and the information required to contact the manufacturer authorized signing authority. When a device is powered on in the owner network, it discovers the owner proxy using link local discovery mechanisms and establishes provisional network connectivity. Through this proxy, the device connects to the domain registrar using a provisional (unauthenticated) TLS connection and presents its IDevID for identification. The device then generates a voucher request and sends it to the registrar, which validates the request and contacts the appropriate MASA. The manufacturer verifies the device identity and issues a signed voucher that authorizes the device to join the owner domain. This voucher is relayed back to the device via the registrar, where the device verifies the manufacturer signature and associated freshness information. Upon successful verification, the device imprints by accepting the owner domain certificate conveyed in the voucher as a new trust anchor. If supported by the owner network, the device then enrolls with a local EST server to obtain a locally issued device identifier certificate for ongoing authentication and management. After these steps are completed, the device is able to securely participate in the owner network. With BRSKI, once the required infrastructure such as the registrar, proxy, and manufacturer services is in place, it enables a bootstrapping experience that requires only physical installation, connectivity, and powering up of the device.

* *Knowledge imparted to the device after protocol execution*: After the BRSKI bootstrapping process, the device obtains the CA fingerprint of the owner’s network domain, which establishes the basis for trusting future interactions with the owner network. If enrollment is performed, the device additionally obtains a Locally Issued Device Identifier (LDevID) certificate. With this trust anchor and optional local identity in place, the device can securely communicate with domain services, participate in secure routing, and be managed by the owner using domain specific management protocols.

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.

* *Knowledge 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) [RFC9112] 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.

* *Knowledge 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. Thread

Thread Mesh Commissioning Protocol (MeshCoP) [threadcommissioning] enables new Thread devices to securely join an existing Thread mesh network. That is, MeshCoP ensures that the network wide Thread Network Key, which is used to protect all link layer and mesh control communications, is securely transferred to a device that the owner/user intends to add to the mesh. MeshCoP relies on a dedicated Commissioner, which may be implemented as a smartphone application, to guide and authorize the commissioning process. Within the Thread mesh, one node acts as the Leader and is responsible for coordinating network wide state. Devices that wish to act as the active Commissioner must petition the Leader for this role. A new device seeking to join the network, called a Joiner, does not yet possess network credentials and therefore relies on an existing mesh node, called a Joiner Router, to relay commissioning traffic. A Border Router hosts a Border Agent and provides connectivity between the IEEE 802.15.4 Thread mesh and external IP networks such as Wi-Fi or Ethernet, enabling interaction with external Commissioners.

Before authorizing Joiners, the Commissioner must first authenticate its own authority to the Thread network. This is achieved by establishing a mutually authenticated DTLS J-PAKE session with the Border Agent using the network寰洉 commissioning credential, known as the PSKc. The PSKc is a key derived from a human-readable passphrase (the Commissioning Credential) chosen by the user during the initial network setup and is typically entered by the user into the Commissioner user interface. Following successful authentication, the Commissioner petitions the network Leader to become the active Commissioner for the mesh network partition. Once accepted, the Commissioner is authorized to manage commissioning and the network begins accepting join attempts for specific Joiner devices that have been enabled by the user.

A new Joiner device scans available radio channels and listens for Discovery Response messages transmitted by nearby routers, which advertise commissioning support. The owner/user facilitates the joining device's entry by providing its unique PSKd printed on the

device or its packaging. Using this PSKd, the Commissioner and the Joiner initiate a DTLS J-PAKE handshake that is relayed through a nearby Joiner Router and the Border Agent to reach the Commissioner. Upon successful authentication, a secure Joiner Session is established and the Commissioner performs the Joiner Entrust phase, during which it securely delivers the Thread Network Key and other operational parameters, such as the Mesh Local Prefix, protected by a session derived Key Encryption Key. After receiving these credentials, the Joiner terminates the commissioning session and uses its newly acquired Network Key to perform the Mesh Link Establishment attach procedure with a parent router, thereby becoming a fully authenticated and operational node in the Thread mesh network.

Thread has the following characteristics:

* *Terms*:

  - _Commissioning_: refers to the entire process by which a new device is authorized to join an existing Thread network. It encompasses the discovery of commissioning-capable networks, authentication of the Commissioner, authentication of the Joiner using a device-specific PSKd, and the secure delivery of network credentials to the Joiner.

  - _Petitioning_: refers specifically to the process by which a prospective Commissioner requests authority from the Thread network to act as the active Commissioner.

  - _Discovery_: used for the mechanisms by which Joiner devices identify Thread networks that are currently accepting new devices.

  - _Provisioning_: refer narrowly to the act of delivering credentials to a Joiner once it has been authenticated.

  - _Entrust_: protocol step in which the Commissioner securely provides the Joiner with the Thread Network Key and other essential network parameters. Entrusting marks the transition point at which the joining device gains the cryptographic material required to participate as a trusted member of the mesh.

  - _Attach_: refers to the process that follows commissioning, in which the newly provisioned device uses the Network Key to perform Mesh Link Establishment (MLE) with a parent router.

  - _Join_: act of a device that is not yet a member of the Thread mesh network joining the mesh network

* *Players*: The device manufacturer is responsible for provisioning each device with a unique EUI-64 identifier and a Pre-Shared Key for the Device (PSKd), making this credential available to the user via physical labels or packaging. The device owner/user is responsible for establishing the Thread network infrastructure prior to commissioning. This includes deploying a functioning Thread mesh with at least one Border Router and selecting a human-readable Commissioning Credential (passphrase). This passphrase is used to derive the PSKc (Pre-Shared Key for the Commissioner), which allows the Commissioner to authenticate and communicate securely with the network's Border Agent. Additionally, the user must ensure an existing Thread device is in the vicinity of the new Joiner to serve as the Joiner Router relay. The Commissioner role may be fulfilled by functionality built into mobile operating systems (Android/iOS) or via a manufacturer-supplied application. Once the infrastructure is ready, the user is responsible for inputting the new device's PSKd into the Commissioner to initiate the mutually authenticated joining process.

* *Initial beliefs assumed in the device*: Thread mesh commissioning assumes that a device is manufactured with a unique EUI-64 and a Pre Shared Key for the Device (PSKd), which is a short human-readable string intended to be conveyed out-of-band to an authorized commissioner. This PSKd is typically printed on the device itself, on an attached label, or on the original packaging, and may also be encoded in a QR code to facilitate user entry. The commissioning process relies on the assumption that this PSKd is securely transferred by the user to the commissioner, such as a smartphone application, and is used to initiate a mutually authenticated commissioning exchange.

* *Processes*: The Thread mesh commissioning process begins when a user authorized Commissioner, commonly implemented as an application or service associated with a border router, discovers the target Thread network via a Border Agent. In parallel, a prospective device, known as a Joiner, is powered on and placed into joining mode, where it actively scans available IEEE 802.15.4 channels and transmits Discovery Request messages. Existing network nodes that are permitted to assist commissioning, such as routers or router eligible end devices, respond with Discovery Responses that advertise network identifiers and Steering Data, enabling the Joiner to identify a compatible network. Before authorizing new devices, the Commissioner must authenticate itself by establishing a mutually authenticated DTLS session with the Border Agent using the network commissioning credential and by petitioning the network Leader to become the active Commissioner for the partition. Once authorized, and after the user supplies the Joiner specific PSKd to the Commissioner, the Joiner initiates

a DTLS based authentication exchange that is relayed through a nearby Joiner Router and the Border Agent. This exchange uses a password authenticated key exchange to mutually prove possession of the PSKd. Upon successful authentication, a secure commissioning session is established and the Commissioner performs the Joiner Entrust procedure, securely delivering the Thread Network Key and other operational parameters to the Joiner. With these credentials installed, the Joiner terminates the commissioning session and proceeds to attach to the mesh by executing the Mesh Link Establishment protocol with a parent router, thereby completing commissioning and becoming a fully authenticated and operational node in the Thread network.

* *Knowledge imparted to the device after protocol execution*: After completion of the Thread mesh commissioning process, the device possesses all information required to securely participate as a node in the Thread network. This includes information such as the network name, channel, PAN identifier, and security policy. The device is also provided the thread network key, which serves as the root of trust for securing link-layer frames and Mesh Link Establishment (MLE) communications; this allows the device to mutually authenticate neighboring nodes as legitimate members of the same security domain. In addition, the device acquires addressing and identity information (Mesh-Local Endpoint Identifier (ML-EID) and Routing Locator (RLOC)) to facilitate efficient IPv6 routing and forwarding within the network.

3. Comparison

This section presents a comparative analysis of the protocols for initial security setup of IoT devices. While the protocols originate from different standards bodies and target diverse deployment environments, they all address the same fundamental challenge of transitioning a device from a factory default state to a secure and operational state within a specific administrative, network, or ownership domain. The comparison is structured along the terminology used by each protocol, the players involved, the initial beliefs assumed on the device, the process steps required to complete onboarding, and finally, the knowledge imparted to the device after protocol execution. The section also discusses broader implications such as privacy considerations, resilience to supply chain attacks, support for resale and reuse, and the trade offs between user involvement and automation.

3.1. Comparison of terminology

The ten protocols covered in Section 2 highlight the lack of a common lexicon for specifying initial security setup mechanisms for IoT devices. While these protocols broadly aim to achieve the same outcome, namely transitioning a device from a factory default state to an operational and trusted network node, the terminology used to describe this process is fragmented and often inconsistent. Although it is difficult to identify strict patterns, several noteworthy observations emerge from a comparison of the terminology used across the protocols.

IETF protocols such as BRSKI, EAP NOOB, and SZTP predominantly rely on _bootstrapping_ as the umbrella term for the entire process of transitioning a device from a factory default state to an operational state. Notably, SZTP includes the word provisioning in its title but explicitly describes itself as a bootstrapping strategy within the specification. In contrast, Wi-Fi DPP uses _bootstrapping_ more narrowly as only the initial step of establishing trust through mechanisms such as QR codes or NFC. This differs from the IETF usage, where bootstrapping typically encompasses the complete process.

OCF and FDO use _onboarding_ as the umbrella term for the entire initial security setup. This terminology reflects a consumer or enterprise management mindset, where a device is brought on board into an administrative or ownership domain. In both protocols, onboarding is closely tied to the concept of ownership transfer, treating the device as an asset that moves between legal or administrative entities such as manufacturers and owners. It is also worth noting a subtle semantic distinction OCF and FDO define onboarding primarily as a verb describing the process of transitioning the device, whereas SZTP defines _onboarding information_ as a noun referring to the payload, such as boot images, configuration, and scripts, that the device receives.

Bluetooth Mesh and Wi Fi DPP use _provisioning_ as the umbrella term for the entire process. This choice is potentially confusing, as provisioning has traditionally referred to a specific sub step involving the delivery of configuration data or credentials. By elevating provisioning to describe the full lifecycle of initial security setup, these protocols blur the conceptual boundary between establishing trust and configuring operational parameters.

Thread uses the term _commissioning_ to describe entire initial setup process. This term has possible roots in building automation and industrial control systems, where it implies a technician physically installing, verifying, and activating equipment. The choice of commissioning reflects Thread寰洉 emphasis on local installation and user assisted setup within a constrained network environment.

The term _provisioning_ exhibits the most severe semantic overloading across protocols. In Wi Fi DPP and Bluetooth Mesh, provisioning refers to the entire process of adding a device, including discovery, authentication, and key distribution. In Thread and OCF, provisioning is a more narrowly defined step that refers specifically to the delivery of credentials and security related information after authentication or ownership transfer. SZTP uses the term more loosely and often interchangeably with bootstrapping, further contributing to ambiguity.

The term _enrollment_ is predominantly used in protocols that rely on X.509 certificates. EST and BRSKI use enrollment in its strict PKI sense, referring to the process by which a device obtains a certificate from a Certificate Authority. In contrast, Wi Fi DPP uses enrollment more informally to mean authorizing a device to join a network, largely detached from its PKI origins. SZTP uses enrollment in a completely different manner, referring to the device owner enrolling with the manufacturer, which reverses the more common direction of device enrolling into a network or domain.

While strict harmonization of terminology across diverse standards is impractical, this analysis reveals that much of the ambiguity arises when terms describing atomic actions, such as transferring credentials, are conflated with terms describing broader lifecycle phases, such as transitioning a device from a factory state to an operational state. To improve clarity, it is beneficial to adopt clear umbrella terms such as _onboarding_ or _bootstrapping_ to describe the comprehensive process. _Onboarding_ is preferable when emphasizing ownership or administrative control transitions, as seen in OCF and FDO, while _bootstrapping_ is more suitable for network centric state transitions, as in BRSKI and SZTP. Action oriented terms such as _provisioning_ and _authentication_ should be reserved for specific sub phases within the umbrella process. In particular, provisioning should ideally refer to the secure delivery of configuration data or credentials, rather than the entire initial security setup lifecycle.

3.2. Comparison of players

The burden of initial security setup can never truly be eliminated; it is instead shifted between the entities involved. In some protocols, this burden is placed primarily on the user, for example by requiring QR code scanning or PIN entry, while in others it is placed on the manufacturer, for example by requiring the operation of online authorization services. In this section, we compare the protocols based on the players involved and the expectations placed upon them.

One important distinction lies in the role of the manufacturer. At one end of the spectrum are protocols with zero preconfiguration, such as EAP NOOB or Bluetooth Mesh using Just Works, which require minimal manufacturer involvement. These protocols do not demand preinstalled unique credentials or cryptographic secrets. Next are protocols such as Thread, Bluetooth Mesh, and Wi Fi DPP, where the manufacturer responsibility is largely limited to the factory. In these cases, the manufacturer must generate a unique identifier, such as an EUI 64, and provision a static bootstrapping secret or key, such as a PSKd or a bootstrapping public key. This information is typically made available physically on the device or its packaging, for example as a printed label or QR code. OCF similarly requires provisioning a unique device UUID and optionally a manufacturer certificate. Once the device leaves the factory, the manufacturer security role largely ends. At the other end of the spectrum are protocols that require sustained manufacturer involvement. BRSKI, SZTP, and FDO require manufacturers to install specific trust anchors and IEEE 802.1AR IDevID certificates and to operate high availability online services. In BRSKI, this role is fulfilled by the Manufacturer Authorized Signing Authority. In SZTP and FDO, manufacturers issue ownership vouchers and may also operate redirect or rendezvous services. These protocols further require manufacturers to track device serial numbers in order to validate ownership claims, introducing significant logistical and operational complexity.

The user or installer occupies very different positions across the protocols. User centric protocols require active participation, such as scanning QR codes, entering PINs, confirming device identity, or approving commissioning actions. This is common in Wi Fi DPP, Bluetooth Mesh, Thread, and OCF Just Works or Random PIN modes. In contrast, protocols designed for minimal or no user interaction at the device level, such BRSKI, SZTP, and FDO largely front load the user responsibilities to system preparation activities such as configuring backend services and defining policies, after which devices can complete the protocol without further user involvement.

Administrator responsibilities also vary significantly between protocols. Infrastructure centric protocols require the deployment and operation of specific components. BRSKI requires a Registrar and often an EST server. LwM2M requires a Bootstrap Server. FDO requires a Rendezvous Server and an onboarding service. For EAP NOOB, the administrator must configure the AAA infrastructure, such as RADIUS, to route the special NAI noob@eap-noob.arpa to the appropriate application server. In SZTP, the network must support specific DHCP options or DNS records to allow devices to discover bootstrap servers. In BRSKI, a Join Proxy is required to facilitate communication between an unauthenticated pledge and the registrar.

The relationship between owner, user, and administrator is not always clearly delineated. In a home network using Bluetooth Mesh or Thread, the owner, administrator, and user are often the same individual. In contrast, when BRSKI is used for large scale deployments such as ISP provided CPE, the administrator and owner may be the service provider, while the end user has little or no role in the onboarding process beyond physically connecting the device.

Another important dimension of comparison is whether a protocol relies primarily on local actors or remote infrastructure. Protocols such as Thread, Bluetooth Mesh, and Wi Fi DPP are largely local in nature and can operate without continuous Internet connectivity. Others, such as BRSKI, SZTP, EST, and FDO, introduce additional remote players operated by manufacturers or service providers, including certificate authorities, manufacturer authorization services, bootstrap servers, and rendezvous services. EAP NOOB also relies on a remote authentication server, although this server may be colocated with local infrastructure such as an access point. One advantage of EAP based approaches is that devices can communicate securely with remote servers before obtaining full IP connectivity, a capability that is inherent to EAP. A general trend across protocols is the gradual introduction of optional remote components even in traditionally local schemes, such as DPP over TCP or remote provisioning in Bluetooth Mesh, reflecting the practical benefits of centralizing logic and state in backend services.

3.3. Comparison of initial beliefs

Protocols can be grouped into four broad categories based on the type and strength of identity and credentials that are imparted to the device during manufacturing or prior to deployment. The choice of category has significant implications for supply chain complexity, user involvement, scalability, and achievable security properties.

* *The Blank Slate (No Cryptographic Secrets)*: In this category, devices are assumed to have no pre-installed cryptographic secrets or unique credentials. Trust is established entirely through user-assisted mechanisms. Examples include OCF Just Works, OCF Random PIN, and EAP-NOOB. Security in this category relies heavily on the user寰洹 ability to correctly associate the physical device in their possession and complete the procedure securely (e.g. using Just Works in a physically secure environment).

* *Static Symmetric Secrets*: These protocols assume that a shared secret or password is installed at manufacturing time. In Thread, the device holds a unique EUI-64 and a PSKd (Pre-Shared Key for Device). The trust model assumes that possession of the PSKd, typically obtained by reading a label or scanning a QR code, implies authorization to commission the device. In LwM2M Factory or Smartcard modes, a symmetric key known to both the device and the Bootstrap Server is used, enabling automated bootstrapping without direct user interaction.

* *Raw Asymmetric Keys*: Devices in this category possess a unique asymmetric key pair but no certificate signed by a trusted authority. In Wi-Fi DPP, the device holds a bootstrapping key pair and the public key is encoded in a QR code or URI that is transferred to the configurator. Similar approaches are used in Bluetooth Mesh with Static OOB authentication and in LwM2M Raw Public Key (RPK) mode. In these protocols, knowledge of the device public key is sufficient to provision or onboard the device.

* *Certificates and Supporting Infrastructure*: The most infrastructure-intensive category assumes that devices are provisioned during manufacturing with X.509 certificates and corresponding trust anchors. BRSKI, SZTP, FDO, EST, OCF in certificate-based modes, and Bluetooth Mesh v1.1 certificate provisioning fall into this category. In BRSKI, the device must possess an IEEE 802.1AR IDevID certificate and a trust anchor to verify vouchers issued by the Manufacturer Authorized Signing Authority (MASA). SZTP similarly assumes an IDevID and trust anchors for verifying manufacturer-signed ownership vouchers and bootstrap servers. FDO assumes a GUID, an attestation key pair, a hash of the manufacturer public key, rendezvous information, and a device-specific HMAC secret used to validate ownership vouchers. EST assumes an existing credential, commonly an IEEE 802.1AR certificate, to mutually authenticate the TLS session with the EST server. OCF certificate-based modes and Bluetooth Mesh v1.1 support optional manufacturer-installed certificates that enable automated device authentication during ownership transfer or provisioning.

The nature of the initial beliefs assumed on the devices by various protocols have other consequences worth considering:

* *Privacy*: Protocols that expose static identifiers can pose privacy risks. Bluetooth Mesh and Thread often advertise a Device UUID or EUI-64 in the clear during discovery, creating a persistent digital fingerprint that can be passively observed. EAP-NOOB includes a PeerInfo field, which may contain information such as MAC address or device make and model. However, this information is revealed only after the server responds to the initial generic identity noob@eap-noob.arpa, ensuring that disclosure occurs only to a server that actively participates in the EAP exchange rather than continuously via broadcast beacons. Another privacy consideration is protection from the device manufacturer. Vendor-mediated protocols such as FDO, SZTP, and BRSKI achieve convenience and minimal user interaction by requiring the device to contact a manufacturer-operated service such as a Rendezvous Server, MASA, or bootstrap server. As a result, the manufacturer or cloud provider can retain a permanent record of when and from which network a device is onboarded. The owner cannot install the device without the manufacturer becoming aware of its activation. In contrast, local sovereignty protocols operate entirely within the local deployment context and remain invisible to the manufacturer. A user can deploy a device in an isolated environment without the vendor ever learning that it was activated.

* *Vulnerability to label swapping*: Protocols in the static symmetric secret and raw asymmetric key categories rely heavily on the physical integrity of device labels. If an attacker can photograph or copy a QR code in the supply chain or replace it with a malicious one, they may be able to claim the device as soon as it comes online or perform a man-in-the-middle attack. Damage to or loss of the label can also render a device unusable unless the manufacturer retains and can recover the associated public key. If the QR code is not attached to the device itself and the device is resold on a secondary market, the new owner may be unable to onboard it. In some ecosystems, devices cannot be transferred to a new owner without cooperation from the previous owner, leading to devices being resold but effectively unusable.

* *Raw public keys vs. certificates*: Protocols based on raw public keys, such as Wi Fi DPP with Static OOB authentication, prevent passive eavesdropping but can be vulnerable to misbinding attacks [Sethi19]. An attacker can print a QR code containing their own public key and attach it to a different device. If the user scans this code, the protocol may complete successfully with an attacker controlled device. In contrast, certificate based approaches

allow a device to cryptographically prove contextual information such as its manufacturer, model, or serial number, reducing the risk that initial security setup is completed with the wrong or malicious device. Both raw public keys and certificates introduce cost factors, since factory installed credentials require secure manufacturing processes to ensure that keys and certificates are generated, stored, and injected without leakage. While the use of certificates is becoming increasingly common and provides the additional benefit of signed contextual information, the complexity and cost of operating a public key infrastructure cannot always be ignored by manufacturers. Commercial private PKI services, such as those offered by cloud providers like Amazon Web Services (AWS), typically charge per issued certificate, which may be prohibitive for certain types of devices.

3.4. Comparison of processes

All protocols solve the same sequence of problems: discovering a device or service endpoint, establishing initial trust, authorizing the device for a specific domain or owner, and delivering the credentials and configuration required for operational use. The protocols differ primarily in where these steps occur, who participates in them, and how much user involvement is required.

A first distinguishing factor is *discovery*. Local and user centric protocols such as Bluetooth Mesh, Thread, and Wi Fi DPP rely on link local discovery mechanisms, including beaconing, active scanning, or out-of-band identifiers such as QR codes or NFC. In these protocols, discovery is tightly coupled to physical proximity and user intent. In contrast, infrastructure centric protocols such as SZTP, BRSKI, FDO, and LwM2M rely on network based discovery mechanisms. Devices locate bootstrap services using DHCP options, DNS records, well known URLs, or rendezvous services. EAP NOOB occupies a hybrid position, where the device uses a well-known identity to trigger bootstrapping with any network willing to route EAP exchange.

The *establishment of initial trust* is another major point of divergence. User assisted protocols typically rely on out-of-band mechanisms such as PINs, QR codes, shared secrets, or visual confirmation to authenticate the first exchange. Examples include Thread PSKd entry, Bluetooth Mesh OOB authentication, Wi-Fi DPP QR code scanning, and OCF Random PIN. These approaches trade automation for user visibility and local control. Infrastructure centric protocols instead rely on manufacturer installed credentials and third-party validation. In BRSKI, the device presents an IDevID certificate and receives a voucher validated by the manufacturer. In SZTP and FDO, ownership vouchers and manufacturer signatures play a similar role. EST assumes the existence of a credential that allows

immediate mutual authentication with a server. In these protocols, trust is anchored in a PKI or manufacturer operated service rather than in user mediated actions.

Protocols also differ in *where authorization decisions are made*. In network centric schemes such as Thread, Bluetooth Mesh, and DPP, authorization is implicit in possession of shared credentials or group membership. Once a device demonstrates knowledge of the appropriate key material, it is accepted as part of the network. In contrast, owner centric protocols such as FDO, OCF, and LwM2M explicitly model ownership and administrative domains. Authorization involves verifying ownership artifacts, registering devices with backend services, or associating them with specific management endpoints. BRSKI similarly requires explicit authorization through voucher validation, binding the device to a specific domain before it is allowed to enroll for credentials.

The *delivery of credentials and configuration* also varies in scope and timing. Some protocols deliver only the minimum information required for secure communication, such as a network key or access credential, leaving higher level configuration to later stages. Thread and Bluetooth Mesh are examples of this approach. Other protocols deliver a richer set of information during onboarding itself. SZTP may deliver boot images, configuration files, and scripts. FDO, OCF, and LwM2M provision management service endpoints, access control parameters, and long-term credentials as part of the onboarding flow. In these protocols, initial security setup is tightly coupled with lifecycle management.

Finally, the protocols differ in their *support for repetition, reset, and ownership transfer*. Some protocols treat onboarding as a one time event, with reset and reprovisioning possible but not explicitly modeled. Thread and Bluetooth Mesh fall largely into this category. Others explicitly design for reuse and transfer. FDO and OCF include mechanisms to update internal credentials and ownership state, enabling resale or reassignment without manufacturer involvement. These differences have significant implications for device longevity, secondary markets, and operational flexibility.

3.5. Comparison of knowledge imparted to the device

Although all surveyed protocols aim to securely transition a device into an operational state, they differ significantly in what beliefs are imparted to the device once the protocol has completed. These post execution beliefs define not only how the device authenticates and communicates afterwards, but also how it interprets authority, ownership, and its position within a broader administrative or security domain.

* *Network centric*: Protocols focused primarily on connectivity impart, at minimum, the cryptographic material required to communicate at the data link or network layer. The device learns how to communicate securely with its neighbors, but not necessarily who manages it or what its long term application behavior should be. For example, Thread imparts the Thread Network Key along with network parameters such as PAN ID and channel, and provides addressing information including ML EID and RLOC, enabling the device to participate in IPv6 routing. Bluetooth Mesh similarly imparts a Network Key and a unicast address, allowing the device to encrypt and authenticate mesh traffic. Wi Fi DPP imparts a Connector, a signed credential containing a group identifier and a network access key, and also provisions the configurator public signing key, which functions as a trust anchor for verifying other network participants. In these protocols, trust is implicitly placed in peers that demonstrate possession of the same credentials. They generally do not encode an explicit concept of ownership, and authorization is derived from membership in a network or group. Bluetooth Mesh partially blurs this boundary by also provisioning Application Keys and a Device Key, enabling continued security provisioning and application level protection beyond basic network access.

* *Infrastructure centric*: Protocols in this category focus on imparting trust anchors and embedding the device into a public key infrastructure domain. Examples include BRSKI, EST, and SZTP. These protocols impart the belief that the device is now a member of a specific PKI domain and that trust relationships are mediated through certificates and certificate authorities. After execution, the device typically holds a locally issued certificate, trust anchors for the domain, and in some cases vouchers or ownership related artifacts that bind its identity to the domain. Authorization decisions are derived from PKI membership and policy rather than from direct user interaction or shared secrets. SZTP goes further than most by also imparting onboarding information that can include boot images, configuration scripts, and software updates. In this case, the knowledge imparted to the device is not limited to credentials, but can extend to defining the complete executable and operational personality of the device.

* *Service centric*: Protocols such as FDO, OCF, LwM2M, and EAP NOOB impart the belief that the device belongs to a specific owner or administrative entity and should report to a specific service endpoint. Upon completion, the device holds an Owner Credential or ownership related trust anchor that cryptographically binds it to an owner or management domain. This belief is stronger and more persistent than simple network membership and enables fine

grained authorization, delegation, and lifecycle management. Rather than binding the device solely to a network, these protocols bind it to a logical service or platform. Crucially, they impart service knowledge: the device completes the protocol knowing exactly which URL or service endpoint to contact for configuration, credential updates, software updates, and access control decisions. In protocols such as OCF and FDO, the device is explicitly provisioned with the identities and endpoints of management services, including Credential Management Services and Access Management Services. The device therefore believes not only who owns it, but also where to obtain policies, credentials, and updates throughout its operational lifetime.

The extent to which a device is prepared for reuse and ownership transfer also varies significantly across protocols. Protocols such as FDO and OCF explicitly update internal state and credentials to support future resale or re-execution of the protocol for initial security setup, imparting the belief that ownership and trust relationships are mutable over time. By contrast, Thread and Bluetooth Mesh generally treat commissioning as a one time event, with reset and reprovisioning possible but not explicitly modeled as ownership transitions. A particularly challenging case arises in protocols that rely on physical labels. If possession of a label or QR code is sufficient to claim a device, then an unauthorized individual with temporary access to the device or its packaging may be able to take control of it. In other cases, a device reset may not be sufficient to make the device reusable until the previous owner explicitly removes it from their management system. These differences have significant implications for secondary markets, device longevity, and long term sustainability.

3.6. Other observations

TODO Discuss several protocols require temporary connectivity via border router authenticator etc. Even DPP and Bluetooth support provisioner not in radio coverage

4. Security Considerations

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

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

[bt-mesh] "Mesh Protocol", v1.1 , 2023, <https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/MshPRT_v1.1/out/en/index-en.html>.

[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/>.

[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>.

[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>.

[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>.

[RFC7616] Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP Digest Access Authentication", RFC 7616, DOI 10.17487/RFC7616, September 2015, <https://www.rfc-editor.org/rfc/rfc7616>.

[RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/rfc/rfc7617>.

[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>.

[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>.

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/rfc/rfc9112>.

[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>.

[Sethi19] Sethi, M., Peltonen, A., and T. Aura, "Misbinding Attacks on Secure Device Pairing and Bootstrapping", Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), pp. 453-464, Auckland, New Zealand, July 2019, <https://doi.org/10.1145/3321705.3329813>.

[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 Specification", Version: 1.4.0 , 2024, <https://www.threadgroup.org/ThreadSpec>.

Authors' Addresses

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

Behcet Sarikaya
Unaffiliated
Email: sarikaya@ieee.org

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