Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 22, 2021

M. Sethi
Ericsson
B. Sarikaya
Denpel Informatique
D. Garcia-Carrillo
University of Oviedo
February 18, 2021

Secure IoT Bootstrapping: A Survey
draft-sarikaya-t2trg-sbootstrapping-11

Abstract

This draft provides an overview of the various terms that are used when discussing bootstrapping of IoT devices. We document terms that have been used within the IETF as well as other standards bodies. We investigate if the terms refer to the same phenomena or have subtle differences. We provide recommendations on the applicability of terms in different contexts. Finally, this document presents a survey of secure bootstrapping mechanisms available for smart objects that are part of an Internet of Things (IoT) network. The survey does not prescribe any one mechanism and rather presents IoT developers with different options to choose from, depending on their use-case, security requirements, and the user interface available on their IoT devices.

Table of Contents

1. Introduction

We informally define bootstrapping as "any process that takes place before a device can become operational". While bootstrapping is necessary for all computing devices, until recently, most of our devices typically had sufficient computing power and user interface (UI) for ensuring somewhat smooth operation. For example, a typical laptop device required the user to setup a username/password as well as enter network settings for Internet connectivity. Following these steps ensured that the laptop was fully operational.

The problem of bootstrapping is however exacerbated for Internet of Things (IoT) networks. The size of an IoT network varies from a couple of devices to tens of thousands, depending on the application. Smart objects/things/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 problem of bootstrapping in IoT was identified by Sethi et al. [Sethi14] while developing a bootstrapping solution for smart displays. Although this document focuses on bootstrapping terminology and methods for IoT devices, we do not exclude bootstrapping related terminology used in other contexts.

Bootstrapping devices typically also involves providing them with some sort of network connectivity. Indeed, the functionality of a disconnected device is rather limited. Bootstrapping 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 passpharse before new devices can be bootstrapped. 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 bootstrapping devices and excludes any discussion on setting up networks before devices can be bootstrapped.

In addition to our informal definition, it is helpful to look at other definitions of bootstrapping. The IoT@Work project defines bootstrapping in the context of IoT as "the process by which the state of a device, a subsystem, a network, or an application changes from not operational to operational" [iotwork]. Vermillard [vermillard] , on the other hand, describes bootstrapping as the procedure by which an IoT device gets the URLs and secret keys for reaching the necessary servers. Vermillard notes that the same process is useful for re-keying, upgrading the security schemes, and for redirecting the IoT devices to new servers.

There are several terms that have often been used in the context of bootstrapping:

- Bootstrapping
- Provisioning
- Onboarding
- Enrollment
- Commissioning
- Initialization
- Configuration
- Registration

We attempt to find out whether all these terms refer to the same phenomena. We begin by looking at how these terms have been used in various standards and standardization bodies in Section 3. We then summarize our understanding in Section 4, and provide our recommendations on their usage in Section 5. Section 6 provides a taxonomy of bootstrapping methods and Section 7 categorizes methods according to the taxonomy.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174].

3. Usage of bootstrapping terminology in standards

To better understand bootstrapping related terminology, let us first look at the terms used by some existing specifications:

3.1. Device Provisioning Protocol (DPP)

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

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

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

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

3.2. Open Mobile Alliance (OMA) Lightweight M2M (LwM2M)

The OMA LwM2M specification [oma] defines an architecture where a new device (LwM2M client) contacts a Bootstrap-server which is responsible for "provisioning" essential information such as credentials. After receiving this essential information, the LwM2M client device "registers" itself with one or more LwM2M Servers which will manage the device during its lifecycle. The current standard defines the following four bootstrapping modes:

- Factory Bootstrap: An IoT device in this case is configured with all the necessary bootstrap information during manufacturing and prior to its deployment.

- Bootstrap from Smartcard: An IoT device retrieves and processes all the necessary bootstrap data from a Smartcard.

- Client Initiated Bootstrap: This mode provides a mechanism for an IoT client device to retrieve the bootstrap information from a Bootstrap Server. This requires the client device to have an account at the Bootstrap Server and credentials to obtain the necessary information securely.

- Server Initiated Bootstrap: In this bootstrapping mode, the bootstrapping server configures all the bootstrap information on the IoT device without receiving a request from the client. This means that the bootstrap server needs to know if a client IoT Device is ready for bootstrapping before it can be configured.

For example, a network may inform the bootstrap server of a new connecting IoT client device.

3.3. Open Connectivity Foundation (OCF)

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

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

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

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

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

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

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

The OBT additionally configures/provisions information about the Access Management Service (AMS), the Credential Management Service (CMS), and the credentials for interacting with them. The AMS is

responsible for provisioning access control entries, while the CMS provisions security credentials necessary for device operation.

3.4. Bluetooth

Bluetooth mesh provisioning. Beacons for discovery. Public-key exchange followed by authentication. Finally provisioning of the network key and unicast address. To be expanded.

3.5. Fast IDentity Online (FIDO) alliance

The Fast IDentity Online Alliance (FIDO) is currently specifying 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 lifecyle. The draft specification refers to this feature as "late binding".

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

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 Restricted Operation Environment (ROE) on the device. The initial information embedded also includes a unique device identifier (called as 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 used.

3.6. Internet Engineering Task Force (IETF)

In this section, we will look at some IETF standards and draft specifications related to IoT bootstrapping.

3.6.1. 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 [I-D.ietf-ace-coap-est]. EST assumes that some initial information is already distributed so that EST client and servers can 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.

3.6.2. Bootstrapping Remote Secure Key Infrastructures (BRSKI)

The ANIMA working group is working on a bootstrapping solution for devices that relies on 802.1AR vendor certificates called Bootstrapping Remote Secure Key Infrastructures (BRSKI) [I-D.ietf-anima-bootstrapping-keyinfra]. In addition to vendor installed IEEE 802.1AR certificates, a vendor based service on the Internet is required. Before being authenticated, a new device only needs link-local connectivity, and does not require a routable address. When a vendor provides an Internet based service, devices can be forced to join only specific domains. The document highlights that the described solution is aimed in general at non-constrained (i.e. class 2+ defined in [RFC7228]) devices operating in a non-challenged network. It claims to scale to thousands of devices located in hostile environments, such as ISP provided CPE devices which are drop-shipped to the end user.

3.6.3. Secure Zero Touch Provisioning

[RFC8572] defines a bootstrapping strategy for enabling devices to securely obtain all the configuration information with no installer input, beyond the actual physical placement and connection of cables. Their goal is to enable a secure NETCONF [RFC6241] or RESTCONF

[RFC8040] connection to the deployment specific network management system (NMS). This bootstrapping method requires the devices to be configured with trust anchors in the form of X.509 certificates. [RFC8572] is similar to BRSKI based on [RFC8366].

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

EAP-NOOB [I-D.ietf-emu-eap-noob] defines an EAP method where the authentication is based on a user-assisted out-of-band (OOB) channel between the server and peer. 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. This method claims to be more generic than most ad-hoc bootstrapping solutions in that it supports many types of OOB channels. The exact in-band messages and OOB message contents are specified and not the OOB channel details. EAP-NOOB also supports IoT devices with only output (e.g. display) or only input (e.g. camera). It makes combined use of both secrecy and integrity of the OOB channel for more robust security than the ad-hoc solutions.

4. Comparison

There are several stages before a device becomes fully operational. This typically involves establishing some initial trust after which credentials and other parameters are configured. For DPP, bootstrapping is the first step before authentication and provisioning of credentials can occur. For EST, bootstrapping happens as the first step when the client devices have no certificates available for starting enrollment. Provisioning/ configuring are terms used for providing additional information to devices before they are fully operational. For example, credentials are provisioned onto the device. But before credential provisioning, a device is bootstrapped and authenticated. Some protocols may only deal with parts of the process. For example, TLS maybe used for authentication after bootstrapping. A separate device management protocol then may run over this TLS tunnel for provisioning operational information and credentials.

5. Recommendations

- It is recommended that the IETF use the term "bootstrapping" for the initial (authentication) step that a device must perform. Bootstrapping will likely happen before the device has obtained full network connectivity.

- It is recommended to use the term "provisioning"/"configuring" for the process of providing necessary information to a device to

become operational after initial authentication is complete. As is evident from above, provisioning and configuring may include bootstrapping and authentication as a sub protocol.

- IETF specifications should aim to avoid mixing terminology or adding new terminology for better consistency.

6. Classification of available mechanisms

Given the large number of bootstrapping protocols and related specifications, it can be helpful to classify them. We categorize the available bootstrapping solutions into the following major classes:

- Managed methods: These methods rely on pre-established trust relations and authentication credentials. They typically utilize centralized servers for authentication, although several such servers may join to form a distributed federation. Example methods include Extensible Authentication Protocol (EAP) [RFC3748], Generic Bootstrapping Architecture (GBA) [TS33220], Kerberos [RFC4120], Bootstrapping Remote Secure Key Infrastructures (BRSKI) and vendor certificates [vendorcert]. EAP Transport Layer Security EAP-TLS [I-D.ietf-emu-eap-tls13] for instance assumes that both the client and the server have certificates to authenticate each other. Based on this authentication, the server authorizes the client for network access. The Eduroam federation [RFC7593] uses a network of such servers to support roaming clients.

- Opportunistic and leap-of-faith methods: In these methods, rather than verifying the initial authentication, the continuity of the initial identity or connection is verified. Some of these methods assume that the attacker is not present during the initial setup. Example methods include Secure Neighbor Discovery (SEND) [RFC3971] and Cryptographically Generated Addresses (CGA) [RFC3972], Wifi Protected Setup (WPS) push button [wps], and Secure Shell (SSH) [RFC4253].

- Peer-to-Peer (P2P) and Ad-hoc methods: These bootstrapping methods do not rely on any pre-established credentials. Instead, the bootstrapping protocol results in credentials being established for subsequent secure communication. Such bootstrapping methods typically perform an unauthenticated Diffie-Hellman exchange [dh] and then use an out-of-band (OOB) communication channel to prevent a man-in-the-middle attack (MitM). Various secure device pairing protocols fall in this category. Based on how the OOB channel is used, the P2P methods can be further classified into two sub categories:

    * Key derivation: Contextual information received over the OOB channel is used for shared key derivation. For example, [proximate] relies on the common radio environment of the devices being paired to derive the shared secret which would then be used for secure communication.

    * Key confirmation: A Diffie-Hellman key exchange occurs over the insecure network and the established key is used to authenticate with the help of the OOB channel. For example, Bluetooth simple pairing [SimplePairing] use the OOB channel to ask the user to compare pins and approve the completed exchange.

- Hybrid methods: Most deployed methods are hybrid and use components from both managed and ad-hoc methods. For instance, central management may be used for devices after they have been registered with the server using ad-hoc registration methods.

It is important to note here that categorization of different methods is not always easy or clear. For example, all the opportunistic and leap-of-faith methods become managed methods after the initial vulnerability window. The choice of bootstrapping method used for devices depends heavily on the business case. Questions that may govern the choice include: What third parties are available? Who wants to retain control or avoid work? In each category, there are many different methods of secure bootstrapping available. The choice of the method may also be governed by the type of device being bootstrapped.

7. IoT Device Bootstrapping Methods

In this section we look at additional bootstrapping protocols for IoT devices which are not covered in Section 3. Protocols already covered in Section 3 however are mentioned in their respective classes. This list is non-exhaustive.

7.1. Managed Methods

EAP-TLS is a widely used EAP method for network access authentication [I-D.ietf-emu-eap-tls13]. It requires certificate-based mutual authentication and a public key infrastructure. The ZigBee Alliance has specified an IPv6 stack for IEEE 802.15.4 [IEEE802.15.4] devices used in smart meters developed primarily for SEP 2.0 (Smart Energy Profile) application layer traffic [SEP2.0]. The ZigBee IP stack uses EAP-TLS for secure bootstrapping of devices.

EAP-PSK [RFC4764] is another EAP method that realizes mutual authentication and session key derivation using a Pre-Shared Key

(PSK). Given the light-weight nature of EAP-PSK, it can be suitable for resource-constrained devices. However, secure distribution of a large number of PSKs can be challenging.

CoAP-EAP [I-D.marin-ace-wg-coap-eap] defines a bootstrapping service for IoT. The authors propose transporting EAP over CoAP [RFC7252] for the constrained link, and communication with AAA infrastructures in the non-constrained link. While the draft discusses the use of EAP-PSK, the authors claim that they are specifying a new EAP lower layer and any EAP method which results in generation is suitable.

Protocol for Carrying Authentication for Network Access (PANA) [RFC5191] is a network layer protocol with which a node can authenticate itself to gain access to the network. PANA does not define a new authentication protocol and rather uses EAP over User Datagram Protocol (UDP) for authentication.

Colin O'Flynn [I-D.oflynn-core-bootstrapping] proposes the use of PANA for secure bootstrapping of resource constrained devices. He demonstrates how a 6LowPAN Border Router (PANA Authentication Agent (PAA)) can authenticate the identity of a joining constrained device (PANA Client). Once the constrained device has been successfully authenticated, the border router can also provide network and security parameters to the joining device.

Hernandez-Ramos et al. [panaiot] also use EAP-TLS over PANA for secure bootstrapping of smart objects. They extend their bootstrapping scheme for configuring additional keys that are used for secure group communication.

Generic Bootstrapping Architecture (GBA) is another bootstrapping method that falls in centralized category. GBA is part of the 3GPP standard [TS33220] and is based on 3GPP Authentication and Key Agreement (3GPP AKA). GBA is an application independent mechanism to provide a client application (running on the User equipment (UE)) and any application server with a shared session secret. This shared session secret can subsequently be used to authenticate and protect the communication between the client application and the application server. GBA authentication is based on the permanent secret shared between the UE's Universal Integrated Circuit Card (UICC), for example SIM card, and the corresponding profile information stored within the cellular network operator's Home Subscriber System (HSS) database. [I-D.sethi-gba-constrained] describes a resource-constrained adaptation of GBA for IoT.

The four bootstrapping modes specified by the Open Mobile Alliance (OMA) Light-weight M2M (LwM2M) standard require some sort of pre-

provisioned credentials on the device. All the four modes are examples of managed bootstrapping methods.

The Kerberos protocol [RFC4120] is a network authentication protocol that allows several endpoints to communicate over an insecure network. Kerberos relies on a symmetric cryptography scheme and requires a trusted third party, that guarantees the identities of the various actors. It relies on the use of "tickets" for nodes to prove identity to one another in a secure manner. There has been research work on using Kerberos for IoT devices [kerberosiot].

It is also important to mention some of the work done on implicit certificates and identity-based cryptographic schemes [himmo], [implicit]. While these are interesting and novel schemes that can be a part of securely bootstrapping devices, at this point, it is hard to speculate on whether such schemes would see large-scale deployment in the future.

7.1.1. Bootstrapping in LPWAN

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

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

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

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

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

From the above, it is clear that all LPWAN technologies rely on pre-provisioned credentials for authentication between a new device and the network. Thus, all of them can be categorized as managed bootstrapping methods.

7.2. Peer-to-Peer or Ad-hoc Methods

While managed methods are viable for many IoT devices, they may not be suitable or desirable in all scenarios. All the managed methods assume that some credentials are provisioned into the device. These credentials may be in the device micro-controller or in a replaceable smart card such as a SIM card. The methods also sometimes assume that the manufacturer embeds these credentials during the device manufacture on the factory floor. However, in many cases the manufacturer may not have sufficient incentive to do this. In other scenarios, it may be hard to completely trust and rely on the device manufacturer to securely perform this task. Therefore, many times, P2P or Ad-hoc methods of bootstrapping are used. We discuss a few example next.

P2P or ad-hoc bootstrapping methods are used for establishing keys and credential information for secure communication without any pre-provisioned information. These bootstrapping mechanisms typically rely on an out-of-band (OOB) channel in order to prevent man-in-the-middle (MitM) attacks. P2P and ad-hoc methods have typically been used for securely pairing personal computing devices such as smart phones. [devicepairing] provides a survey of such secure device pairing methods. Many original pairing schemes required the user to enter the same key string or authentication code to both devices or to compare and approve codes displayed by the devices. While these methods can provide reasonable security, they require user interaction that is relatively unnatural and often considered a nuisance. Thus, there is ongoing research for more natural ways of pairing devices. To reduce the amount of user-interaction required in the pairing process, several proposals use contextual or location-dependent information, or natural user input such as sound or movement, for device pairing [proximate].

The local association created between two devices may later be used for connecting/introducing one of the devices to a centralized server. Such methods would however be classified as hybrids.

EAP-NOOB [I-D.ietf-emu-eap-noob] is an example of P2P and ad-hoc bootstrapping method that establishes a security association between an IoT device (node) and an online server (unlike pairing two devices for local connections over WiFi or Bluetooth).

Thread Group commissioning [threadcommissioning] introduces a two phased process i.e. Petitioning and Joining. Entities involved are leader, joiner, commissioner, joiner router and border router. Leader is the first device in Thread network that must be commissioned using out-of-band process and is used to inject correct user generated Commissioning Credentials (can be changed later) into Thread Network. Joiner is the node that intends to get authenticated and authorized on Thread Network. Commissioner is either within the Thread Network (Native) or connected with Thread Network via a WLAN (External).

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

7.3. Leap-of-faith/Opportunistic Methods

Bergmann et al. [simplekey] develop a secure bootstrapping mechanism that does not rely on pre-provisioned credentials using resurrecting-duckling imprinting scheme. Their bootstrapping protocol involves three distinct phases: discover (the duckling node searches for network nodes that can act as mother node), imprint (the mother node imprints a shared secret establishing a secure channel once a positive response is received for the imprinting request) and configure (additional configuration information such as network prefix and default gateway are configured). In this model for bootstrapping, a small initial vulnerability window is acceptable and can be mitigated using techniques such as a Faraday Cage (securing the communication physically) to protect the environment of the mother and duck nodes, though this may be inconvenient for the user.

7.4. Hybrid Methods

[RFC7250] defines how raw public keys can be used for mutual authentication of devices and servers. The extension specified in [RFC7250] simplifies client_certificate_type and server_certificate_type to carry only SubjectPublicKeyInfo structure with the raw public key instead of many other parameters found in typical X.509 version 3 certificates. Each side validates the keys received with pre-configured values stored. Using raw public keys for bootstrapping can be seen as a hybrid method. This is because it generally requires an out-of-band (OOB) step (P2P/Ad-hoc) where the raw public keys [RFC7250] are provided to the authenticating entities, after which the actual authentication occurs online (managed). CoAP already provides support for using raw public keys (see Section 9.1.3.2. of [RFC7252])

8. Security Considerations

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

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

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

Secondly, re-bootstrapping of IoT devices may be required since keys have limited lifetimes and devices may be lost or resold. Protocols and systems must have adequate provisions for revocation and re-bootstrapping. Re-bootstrapping must be as secure as the initial bootstrapping regardless of whether this re-bootstrapping is done manually or automatically over the network.

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

mechanisms for identifying and distributing keys to the current member devices of each group.

9. IANA Considerations

There are no IANA considerations for this document.

10. 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.

11. Informative References

[devicepairing] Mirzadeh, S., Cruickshank, H., and R. Tafazolli, "Secure Device Pairing: A Survey", IEEE Communications Surveys and Tutorials , pp. 17-40, 2014.

[dh] Diffie, W. and M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory , vol. 22, no. 6, pp. 644-654, 1976.

[dpp] Wi-Fi Alliance, "Wi-Fi Device Provisioning Protocol (DPP)", Wi-Fi Alliance , 2018, <https://www.wi-fi.org/download.php?file=/sites/default/files/private/Device_Provisioning_Protocol_Specification_v1.1_1.pdf>.

[fidospec] Fast Identity Online Alliance, "FIDO IoT Spec", Fido Alliance , August 2020, <https://fidoalliance.org/specs/internet-of-things/FIDO-IoT-spec.html>.

[himmo] Garcia-Morchon, O., Rietman, R., Sharma, S., Tolhuizen, L., and J. Torre-Arce, "DTLS-HIMMO: Efficiently Securing a Post-Quantum World with a Fully-Collusion Resistant KPS", Submitted to NIST Workshop on Cybersecurity in a Post-Quantum World , version 20141225:065757, December 2014, <https://eprint.iacr.org/2014/1008>.

[I-D.ietf-ace-coap-est] Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST over secure CoAP (EST-coaps)", draft-ietf-ace-coap-est-18 (work in progress), January 2020.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-45 (work in progress), November 2020.

[I-D.ietf-emu-eap-noob]
Aura, T., Sethi, M., and A. Peltonen, "Nimble out-of-band authentication for EAP (EAP-NOOB)", draft-ietf-emu-eap-noob-03 (work in progress), December 2020.

[I-D.ietf-emu-eap-tls13]
Mattsson, J. and M. Sethi, "Using EAP-TLS with TLS 1.3", draft-ietf-emu-eap-tls13-13 (work in progress), November 2020.

[I-D.marin-ace-wg-coap-eap]
Marin-Lopez, R. and D. Garcia-Carrillo, "EAP-based Authentication Service for CoAP", draft-marin-ace-wg-coap-eap-07 (work in progress), January 2021.

[I-D.oflynn-core-bootstrapping]
Sarikaya, B., Ohba, Y., Cao, Z., and R. Cragie, "Security Bootstrapping of Resource-Constrained Devices", draft-oflynn-core-bootstrapping-03 (work in progress), November 2010.

[I-D.sethi-gba-constrained]
Sethi, M., Lehtovirta, V., and P. Salmela, "Using Generic Bootstrapping Architecture with Constrained Devices", draft-sethi-gba-constrained-01 (work in progress), February 2014.

[IEEE802.15.4]
"IEEE Std. 802.15.4-2015", April 2016, <http://standards.ieee.org/findstds/standard/802.15.4-2015.html>.

[implicit]
Porambage, P., Schmitt, C., Kumar, P., Gurtov, A., and M. Ylianttila, "Pauthkey: A pervasive authentication protocol and key establishment scheme for wireless sensor networks in distributed iot applications", International Journal of Distributed Sensor Networks , Hindawi Publishing Corporation , 2014.

[iotwork] European Commission FP7, "IoT@Work bootstrapping architecture Deliverable D2.2", June 2011.

[kerberosiot] Hardjono, T., "Kerberos for Internet-of-Things", February 2014, <https://kit.mit.edu/sites/default/files/documents/Kerberos_Internet_of%20Things.pdf>.

[LoRaWAN] Sornin, N., Luis, M., Eirich, T., and T. Kramp, "LoRa Specification V1.0", January 2015, <https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf>.

[ocf] Open Connectivity Foundation, "OCF Security Specification", Open Connectivitiy Foundation , June 2017, <https://openconnectivity.org/specs/OCF_Security_Specification_v1.0.0.pdf>.

[oma] Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification: Core", Open Mobile Alliance , November 2020, <www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf>.

[panaiot] Hernandez-Ramos, J., Carrillo, D., Marin-Lopez, R., and A. Skarmeta, "Dynamic Security Credentials PANA-based Provisioning for IoT Smart Objects", 2nd World Forum on Internet of Things (WF-IoT) , IEEE , 2015.

[proximate] Mathur, S., Miller, R., Varshavsky, A., Trappe, W., and N. Mandayam, "Proximate: proximity-based secure pairing using ambient wireless signals.", Proceedings of MobiSys International Conference , pp. 211-224, June 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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/info/rfc2904>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <https://www.rfc-editor.org/info/rfc3971>.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <https://www.rfc-editor.org/info/rfc3972>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, DOI 10.17487/RFC4120, July 2005, <https://www.rfc-editor.org/info/rfc4120>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

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

[RFC4764] Bersani, F. and H. Tschofenig, "The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method", RFC 4764, DOI 10.17487/RFC4764, January 2007, <https://www.rfc-editor.org/info/rfc4764>.

[RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191, May 2008, <https://www.rfc-editor.org/info/rfc5191>.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <https://www.rfc-editor.org/info/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/info/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/info/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/info/rfc7228>.

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

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[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/info/rfc7252>.

[RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <https://www.rfc-editor.org/info/rfc7593>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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/info/rfc8366>.

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

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/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/info/rfc8949>.

[SEP2.0] ZigBee Alliance, "ZigBee IP Specification", March 2014, <hhttp://www.zigbee.org/non-menu-pages/zigbee-ip-download/>.

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

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

[simplekey] Bergmann, O., Gerdes, S., and C. Bormann, "Simple Keys for Simple Smart Objects", Smart Object Security Workshop, IETF 83 , March 2012.

[SimplePairing] Bluetooth, SIG, "Simple pairing whitepaper", Technical report , 2007.

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

[TS33220] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) (Release 14)", December 2016, <http://www.3gpp.org/DynaReport/33220.htm>.

[vendorcert] IEEE std. 802.1ar-2009, "Standard for local and metropolitan area networks - secure device identity", December 2009.

[vermillard]
Vermillard, J., "Bootstrapping device security with lightweight M2M", Appeared on blog at medium.com , February 2015.

[wps] Wi-Fi Alliance, "Wi-fi protected setup", Wi-Fi Alliance , 2007.

Authors' Addresses

Mohit Sethi
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: mohit@piuha.net

Behcet Sarikaya
Denpel Informatique

Email: sarikaya@ieee.org

Dan Garcia-Carrillo
University of Oviedo
Oviedo 33207
Spain

Email: garciadan@uniovi.es