Abstract

This document presents a survey of secure bootstrapping mechanisms available for smart objects that are part of an Internet of Things (IoT) network. It aims to provide a structured classification of the available mechanisms. The document does not prescribe any one secure bootstrapping 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 smart objects.

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 August 1, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document.
1. Introduction

An Internet of Things (IoT) network consists of connected things that cooperate together to accomplish tasks such as smart buildings, smart environment monitoring system, and intelligent transport systems. The size of an IoT network varies from a couple of devices to tens of thousands depending on the application. A smart object, or a thing, or a device in an IoT network is typically produced by a variety of vendors and are typically heterogeneous in terms of the constraints on their power supply, communication capability, computation capacity and memory available. Due to this heterogeneity, a wide variety of bootstrapping mechanisms are proposed and used for these smart objects.

Before classifying and describing the various methods of bootstrapping, it is important to discuss what is meant by the term bootstrapping. In order to understand the term bootstrapping, we need to discuss some important preliminaries first. We start by discussing the meaning of identity and identifiers. The dictionary defines identity as "something that distinguishes an entity from other entities". Dick Hardt (an advocate of identity 2.0 concept) in his keynote talk on identity describes human identity as "who you are, what you like, what you say about yourself and what others say about you" [identity2.0]. In addition to human beings, other
entities in our physical environment such as the electronic devices we use, our pets and wildlife also have identities.

Just as in the real world, humans also have identities in the digital world. For example, a digital identity may be used by online service to verify the identity of a registered user and provide it with secure personalized service. This process of identity verification is also known as authentication. An attribute that can be used to identify and distinguish one entity from another is referred to as an identifier. The passport number of a citizen is an example of a real-world identifier. Similarly, an email address is an example of a digital identifier. Often the digital identifier of a human user and the digital identifier of its electronic devices are used interchangeably and one may subsume the other for authentication purposes. For instance, when performing network access authentication, the user may enter its identity credentials on the device that should connect to the network. In this case, the device assumes the user identity on its behalf and authenticates to obtain network access. Ubiquitous computing devices increasingly interact with each other without human intervention. This essentially requires the devices to have their own identifiers for authentication and secure communication.

With these preliminaries in mind, we try to decipher the meaning of bootstrapping. The term itself has often been used in many different contexts. For instance, [RFC4640] describes bootstrapping as the process by which a mobile IPv6 node obtains information about the home address, the home agent address, and a security association. The IoT@Work project defines bootstrapping in the context of Internet of Things (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]. [I-D.oflynn-core-bootstrapping] also discusses the problem of secure bootstrapping for resource-constrained devices and highlights the role of IETF in defining suitable solutions.

We define bootstrapping as any process that is required before the resource-constrained device network can operate. Similarly, Vermillard [vermillard] describes bootstrapping as the procedure by which an IoT device gets the secret keys and URL for reaching the necessary servers. Vermillard notes that this procedure is also useful for re-keying, upgrading the security schemes and for redirecting the devices to other servers. The term device onboarding refers to similar ideas and is often used interchangeably with the term bootstrapping. As an example, the AllSeen Alliance [allseen] defines onboarding as a service that provides a common and simple way for new devices to be brought onto an existing WiFi network. Some
solutions and standards organizations distinguish the processes involved in bootstrapping into the following sub-processes:

a. initial establishment of keys and configuration information

b. subsequent provisioning of keys and configuration information

The Open Connectivity Foundation (OCF), for example, uses the term onboarding for (a) and bootstrapping for (b). Some specifications consider (a) out of scope and assume that this information is manufacturer provisioned. Instead of providing yet another definition of bootstrapping, here we list the different goals that bootstrapping may be used to fulfill:

- Authentication of a pre-established identity or creation of a new identity: To illustrate this with an example, consider the case where a user wishes to use one of the many free online mail services. The user in this case needs to first register and create a unique identifier (email address) for its identity. Thereafter, the user will use this email address along with the password to authenticate and access the mails. Both these processes can be considered as a part of bootstrapping.

- Authorization for network access that may include configuration of communication parameters: Bootstrapping also includes the process by which a device authenticates to the network and receives authorization and credentials for subsequent secure communication.

- Registration or joining a domain or group: The process by which a windows device joins a windows domain can also be seen as bootstrapping.

- Pairing with a specific node, or connecting to a cloud service: Securely pairing two personal computing devices that have no a-priori information about each other, and securely connecting a device to an online cloud service are both different forms of bootstrapping.

It is evident that bootstrapping maybe used in many diverse scenarios to fulfill different goals. Thus, it is not surprising that there are many different bootstrapping protocols and methods available. Rather than trying to achieve the impossible target of enlisting all the different bootstrapping solutions, we instead classify them into the following categories in section 3.
2. Terminology

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

Onboarding: Commonly referred to the phase of bootstrapping where the initial establishment of keys and configuration information happens.

Bootstrapping: Subsequent (to onboarding) provisioning of keys and configuration information.

Opportunistic security: Opportunistic security as defined in [RFC7435] assumes cleartext not comprehensive protection, is the default baseline communication security policy. Encryption and authentication negotiated and applied to the communication when they become available. Opportunistic security allows all the traffic to be encrypted even when there is no authentication. It can be used in devices with no memory and no implied trust exists.

Leap-of-faith security: Leap-of-faith (LOF) or Trust on First Use (TOFU) is based on accepting and establishing security associations with peers without authenticating their identities. As such it is a slight enhancement of opportunistic methods.

An opportunistic bootstrapping method for IoT can be for example a Bluetooth sensor that would connect to any phone in the room which wants to control it. A leap-of-faith method would be a Bluetooth sensor that would connect to the first phone it sees in the room and then always connect to it until it is reset. But even this leap-of-faith was opportunistic the first time. It is only after the initial vulnerability that you have some certainty of talking to the same sensor.

3. Classification of available mechanisms

While some bootstrapping approaches are more user-intensive and require extensive user-involvement by scanning QR codes or entering passwords; others maybe more automated, such as those that rely on mobile networks [I-D.sethi-gba-constrained]. We classify available bootstrapping solutions into the following major categories:

- Managed methods: These bootstrapping 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)
Opportunistic and leap-of-faith methods: In these bootstrapping 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].

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.

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. Another example P2P or Ad-hoc method is EAP-NOOB [I-D.aura-eap-noob] that specifies nimble out-of-band authentication for EAP. 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.

It is important to note here that categorization of different bootstrapping 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. Depending on the link-layer technology used, and the User Interface (UI) available, one or more of the above mentioned methods might be suitable.

4. IoT Device Bootstrapping Methods

In this section we look at some of the recent bootstrapping proposals for IoT devices both at the IETF and elsewhere. Needless to say, if the devices are capable in terms of their computation power and UI available, they can always rely on many existing methods such as username and password combinations and various EAP methods.

4.1. Managed Methods

We first discuss some examples of managed bootstrapping methods.

EAP-TLS [RFC7250] is a widely used EAP method for network access authentication. It allows mutual authentication and distributes the keying material for secure subsequent communications. However it only supports certificate-based mutual authentication, and therefore a public key infrastructure is required. The ZigBee Alliance has specified an IPv6 stack aimed at IEEE 802.15.4 [IEEE802.15.4] devices mainly 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. It realizes mutual authentication and session key derivation using a Pre-Shared Key (PSK). Normally four messages are exchanged in the authentication process. Once the authentication is successful, EAP-PSK provides a protected communication channel. Given the light-weight nature of EAP-PSK, it can often be a good choice on constrained devices.

COAP-EAP [I-D.marin-ace-wg-coap-eap] defines a bootstrapping service for IoT. They propose the transport of EAP over CoAP [RFC7252] for the constrained link, and communication with AAA infrastructures in
the non-constrained link to provide scalability among other
characteristics. Upon a successful authentication, key material is
derived to protect CoAP messages exchanged between the smart object
and the authenticator. They discuss the use of EAP-PSK in the draft,
but state that, since they are specifying a new EAP lower layer, any
EAP method that results in generation of cryptographic material 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 also extend their bootstrapping scheme for
configuring additional keys that are used for secure group
communication.

When a device is not a direct neighbor of the authenticator, its
parent node MUST act as relay. Different EAP encapsulation protocols
have different mechanisms for the relay function, such as the PANA
Relay Element (PRE).

After a successful bootstrapping, the device runs neighbor discovery
protocol to get an IPv6 address assigned [RFC6775]. Data transfer
can be secured using DTLS or IPsec. Keys derived from EAP TLS are
used in either generating DTLS ciphering keys after a successful DTLS
handshake or IPsec ESP ciphering keys after a successful IKEv2
handshake.

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 to IoT applications.

Open Mobile Alliance (OMA) Light-weight M2M standard also defines secure bootstrapping for resource-constrained IoT devices with a centralized Bootstrapping Server (BS). 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.

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].

[I-D.kumar-6lo-selective-bootstrap] presents a selective bootstrapping/commissioning method by introducing the concept of Commissioning Tool (CT). In this method the devices are left to connect to the network and execute 6LowPAN neighbor discovery protocol and have an IPv6 address before they are authenticated. Then the devices are selected one by one in some order to communicate with the CT via untrusted constructed route. Once the ID of joining device is authenticated, the CT sends the layer-2 key material to the device via secured channel. This secure channel is established with
DTLS with credential material that has to be installed onto the device during its manufacture.

Before closing the discussion on managed methods, 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.

4.2. Hybrid Methods

The ANIMA working group is also working on a bootstrapping solution for resource-constrained devices that relies on 802.1AR vendor certificates [I-D.ietf-anima-bootstrapping-keyinfra] called Bootstrapping Remote Secure Key Infrastructures (BRSKI). 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.

[I-D.ietf-netconf-zerotouch] 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 [I-D.ietf-netconf-restconf] 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. [I-D.ietf-netconf-zerotouch] is similar to BRSKI based on [RFC8366], but using a different set of assumptions about communications, including none (USB key).

4.3. Bootstrapping in LPWAN

Low Power Wide Area Network (LPWAN) encompasses a wide variety of technologies, generally, with severe constraints in the link in comparison with other typical IoT technologies such as Bluetooth or IEEE 802.15.4. LPWAN typically presents a star topology with support for thousands of devices per antenna.
Among the wide variety of technologies considered as part of LPWAN, we highlight the ones mentioned in the LPWAN overview document of the LPWAN working group [RFC8376]: LoRaWAN, Narrowband IoT (NB-IoT), SIGFOX and Wi-SUN Alliance Field Area Network (FAN). Each technology has different methods to provide security for the communications. Bootstrapping is not directly tackled by all of them, having in some cases proprietary solutions that are not publicly accessible and in other cases key distribution is not even considered. Among the previous LPWAN technologies, bootstrapping is considered in Wi-SUN Alliance Field Area Network (FAN) and LoRaWAN provides Joining process to derive key material based on some previous key material installed in the device.

Following the definition in Section 3 we find that LPWAN bootstrapping technologies described below all fall into the managed classification. This is because in one way or another, a previous trust relationship has been established and authentication credentials have been installed in the devices.

- LoRaWAN [LoRaWAN] describes its own protocol to authenticate their nodes and incorporate them into their network. This process is called the Joining Procedure and it is based on pre-shared keys. The Joining procedure entails one exchange where the node that intends to join the network sends its identity along with other information to authenticate against a network server which interacts with an entity that knows the pre-shared key (called AppKey) and derives the necessary key material for its nominal operation. There is some variation regarding the pre-installed key material on version 1.0 and 1.1, but the Joining Process is very similar in both cases. The Joining Process consists of an exchange of two messages, the Join-request message (sent from the node) where information about the identity of the node is provided and the Join-accept message (received by the node). In this last message the node receives the necessary information to derive the key material to secure the communications. To this process there are adaptations to use AAA infrastructures to enhance the joining process with AAA features such as identity federation. Since there are pre-established trust relationships and authentication credentials, LoRaWAN falls into the managed category.

- Wi-SUN Alliance Field Area Network (FAN) uses an EAP lower layer (IEEE 802.1X) and the EAP-TLS method for network access authentication and performs the 4-way handshake to establish a security association similarly to a WPA2-Enterprise deployment. Since it uses on the authentication protocols which are used to exemplify the managed methods (EAP-TLS), WI-SUN falls in the managed category as well.
NB-IoT supports EAP-AKA to authenticate the node before accessing the core network. NB-IoT also falls into the category of managed methods, since there is a pre-established trust relationship.

Sigfox provides security to the communications using a unique device ID and an cryptographic keys that are independent for each device. As stated in [RFC8376] algorithms and keying details for are not published, but what we can see is that the establishment of the keys are subject to a pre-established trust relationship with the Sigfox network, hence having also a managed method.

In short, LPWANs are still under development, and as it is identified in [RFC8376], due to the characteristics of these technologies, they are prime candidates to benefit from a standardized Authentication, Accounting, and Authorization (AAA) infrastructure [RFC2904] as a way of offering a scalable solution for some of the security and management issues that are present in LPWANs.

4.4. Peer to Peer or Adhoc 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 Adhoc 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 smartphones. [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.aura-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).

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 Internet-of-Things 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. Also, EAP-NOOB 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.

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 (Network Name, Security Policy, Steering Data, Commissioning Data Timestamp, Commissioning Credential, Network Master Key, Network Key Sequence, Network Mesh-Local ULA, Border Router Locator, Commissioner Session ID, XPANID, PANID and Channel)
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 operates 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.

DPP (Device Provisioning Protocol) [dpptech] is a 3 message authentication protocol currently being standardized by the WiFi Alliance for devices that rely on IEEE 802.11 link-layer for communication. The DPP specification allows devices to join a network without a password using various mechanisms such as QR codes or NFC tags. With DPP, devices can perform mutual authentication without a password. Authentication Request, Response and Confirm are 3 messages types based on [IEEE802.11] format. It provides authentication and key establishment between an initiator and a responder. Out of band mechanisms i.e. QR code, USB, NFC, Bluetooth or proof of shared code/phrase/word is used to acquire bootstrapping key [dpptech]. Afterwards, authentication and configuration exchange takes place. Bootstrap trust in public key can be only for responder’s public key or for both parties in mutual authentication manner. The role of initiator and responder as either enrollee or configurator is decided during initial exchange of DPP Authentication frames. Configurator’s protocol key is always a one-time-used (ephemeral) key but enrollee’s protocol key always becomes its network access provisioning key.

4.5. Leap-of-faith/Oppportunistic Methods

Next, we look at a leap-of-faith/opportunistic bootstrapping method for IoT devices.

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.

[RFC7250] defines how raw public keys can be used to authenticate
constrained devices for mutual authentication using EAP-TLS or DTLS.
Raw public key TLS/DTLS extension 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 the certificates. The device and the authentication server
(AS) exchange client_hello and server_hello messages and send their
raw public keys. The device and AS validate the keys by comparing
the pre-configured values [I-D.sarikaya-6lo-bootstrapping-solution].
This bootstrapping method can be seen as a hybrid. 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). Raw public key approach when used with DTLS offers a
simple secure bootstrapping solution especially for smart energy and
building automation applications. It can be easily integrated with
the Constrained Application Protocol (CoAP).

5. Security Considerations

Bootstrapping protocols that do not rely on a pre-shared key for peer
authentication generally rely on an online or offline third-party
(e.g., an authentication server, a key distribution center in
Kerberos, a certification authority in PKI, a private key generator
in ID-based cryptography and so on) to prevent man-in-the-middle
attacks during bootstrapping. Depending on use cases, a resource-
constrained device may not always have access to an online third-
party for bootstrapping. Some bootstrapping methods therefore rely
on a configuration tool (such as a smartphone) that assists the
bootstrapping process by providing temporary reachability to the
online server.

Depending on use cases, a bootstrapping protocol may deal with
authorization separately from authentication in terms of timing and
signaling path. For example, two resource-constrained devices A and
B may perform mutual authentication using authentication credentials
provided by an offline third-party X whereas resource-constrained
device A obtains authorization for running a particular application
with resource-constrained device B from an online third-party Y
before or after the authentication. In some use cases,
authorization is tightly coupled, e.g.,
successful authentication also means successful authorization.

If authorization information communicated includes cryptographic
keys, care must be taken for provisioning the keys, e.g., guidelines
for AAA-based key management are described in [RFC4962]. Re-bootstrapping of IoT devices may required and therefore there must be adequate provisions for revocation and re-bootstrapping of authentication/authorization credentials. Re-bootstrapping must be as secure as the initial bootstrapping regardless of whether this re-bootstrapping is done manually or automatically over the network.

If resource-constrained devices use a multicast group key for authentications of peers that belong to the group, or for message authentication/encryption, the group key must be securely distributed to the current members of the group. Protocols designed for group key management [RFC4046] may be used for group key distribution after the initial bootstrapping. Alternatively, key wrap attributes for securely encapsulating group key may be defined in network access authentication protocols such as PANA. Those protocols use an end-to-end, point-to-point communication channel with a pair-wise security association between a key distribution center and each key recipient. Further considerations may be needed for more efficient group key management to support a large number of resource-constrained devices.

6. IANA Considerations

There are no IANA considerations for this document.

7. Acknowledgements

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

8. Informative References


[I-D.sethi-gba-constrained]

[identity2.0]

[IEEE802.11]

[IEEE802.15.4]

[implicit]

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

[kerberosiot]


Authors’ Addresses

Behcet Sarikaya  
Denpel Informatique  
Email: sarikaya@ieee.org

Mohit Sethi  
Ericsson  
Hirsalantie 11  
Jorvas 02420  
Finland  
Email: mohit@piuha.net

Dan Garcia-Carrillo  
Odin Solutions, S.L  
Murcia 30820  
Spain  
Email: dgarcia@odins.es