Sethi, Mohit; Aura, Tuomas

Secure Network Access Authentication for IoT Devices: EAP Framework vs. Individual Protocols

Published in:
IEEE Communications Standards Magazine

DOI:
10.1109/MCOMSTD.201.2000088

Published: 01/09/2021

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Sethi, M., & Aura, T. (2021). Secure Network Access Authentication for IoT Devices: EAP Framework vs. Individual Protocols. *IEEE Communications Standards Magazine*, *5*(3), 34-39.
https://doi.org/10.1109/MCOMSTD.201.2000088

Secure network access authentication for IoT devices: EAP framework vs. individual protocols

Mohit Sethi, *NomadicLab, Ericsson Research, Finland* and *Aalto University, Finland*
Tuomas Aura, *Aalto University, Finland*

Abstract: Secure bootstrapping of Internet of Things (IoT) devices is often a multi-step process that begins with enabling Internet access through a local wireless network. The process of enabling Internet access on IoT devices includes network discovery and selection, access authentication, and configuration of necessary credentials and parameters. On the one hand, there are many standard protocols available for network access authentication of IoT devices. On the other hand, Extensible Authentication Protocol (EAP) is a standard framework with support for many authentication methods, and it is primarily used for network access authentication in enterprise networks. This article discusses whether the EAP framework is beneficial for network access authentication of IoT devices.

Network access authentication for IoT devices – current state of affairs

IoT devices require Internet connectivity for data processing and storage in the cloud as well as for remote access and control. Additionally, access to the Internet is also critical for patching security vulnerabilities in IoT devices with over-the-air software updates. Enabling Internet access is one of the primary procedures necessary for bootstrapping IoT devices. The bootstrapping process typically involves additional secondary operations such as associating the new device with a user and configuring access control policies. Next, we will look at some protocols currently deployed or standardized for network discovery and access authentication in IoT devices.

Wi-Fi

Many IoT devices such doorbells, printers, refrigerators, surveillance cameras, and televisions reuse the existing Wi-Fi infrastructure for Internet connectivity because of its widespread deployment and low marginal cost for adding new devices. To enable Internet access for such devices, users need to configure the Service Set Identifier (SSID) and passphrase of their access point. However, IoT devices have limited I/O capabilities and therefore users rely on a companion device such as a smartphone for configuring the SSID and passphrase. The configuration process begins with IoT devices broadcasting their own SSID after being powered-up. The device's SSID and an associated secret passphrase maybe printed on the device or on the box in which it is shipped. Users scan for unconfigured devices on their smartphone and select the specific device which they wish to configure from a list. Thereafter, users connect to the device being configured using the printed passphrase and send the target Wi-Fi network credentials (including the SSID name and passphrase). The target Wi-Fi network is the network that the device should eventually use for Internet connectivity.

The Wi-Fi alliance aims to improve the process for configuring SSID and passphrase on IoT devices with their newly standardized Device Provisioning Protocol (DPP) [1]. DPP compliant IoT devices (known as enrollees in DPP terminology) are expected to ship with an asymmetric key pair as well as a method for communicating the public key and other metadata over an out-of-band (OOB) channel such as Near Field Communication (NFC) or a two-dimensional QR code. DPP has three phases. During the first phase, a user with a smartphone (known as the configurator in DPP terminology) obtains the public key of the IoT device over an OOB channel. Thereafter, the

smartphone and the IoT device exchange ephemeral keys and generate a shared secret which is authenticated with the public key of the device (enrollee) communicated to the smartphone (configurator) over the OOB channel. Finally, the smartphone uses the shared secret to send protected configuration information which includes the SSID and passphrase that the IoT device should use for Internet connectivity.

Zigbee Smart Energy

Zigbee is a mesh networking standard specified by the Zigbee alliance for resource-constrained IoT devices. In typical Zigbee networks, a node, known as the coordinator, is responsible for establishing and managing the overall network. This coordinator also acts as the trust center and authenticates new devices joining the mesh network before providing them the network keys. With the introduction of Zigbee 3.0, all Zigbee devices are shipped with installation codes which are also printed on the device or its packaging. For adding a new device, a user sends the installation code of the device to the trust center over an OOB channel. For example, a user may read the installation code printed as a hexadecimal string on the device packaging and type it into the trust center. This installation code is then transformed into a link key by using a hash algorithm. Finally, the link key is used by the trust center to protect the network key and other configuration information sent to the new device. Zigbee Smart Energy (ZSE) [2] is a profile of Zigbee which is widely deployed in smart meters. In ZSE, the initial link key derived from the installation code does not provide full access privileges. The device and the trust center must derive a new link key with implicit certificates (installed during the manufacturing process) and the Certificate Based Key Exchange (CBKE) protocol [3].

LoRaWAN

LoRaWAN [4] is a wireless network technology developed by the LoRa Alliance and it operates in the Industrial, Scientific, and Medical (ISM) band. LoRaWAN is intended for resource-constrained battery-operated devices and it relies on a star-of-stars topology where messages from devices are relayed by gateways to a central network server. LoRaWAN describes its own network access authentication protocol based on pre-shared keys referred to as AppKeys in the standard. The joining procedure comprises only one message exchange (join-request and join-accept) between a new device and the network server.

Narrow Band IoT

Narrow Band IoT (NB-IoT) is a radio technology developed by 3rd Generation Partnership Project (3GPP) and it provides Internet-connectivity to IoT devices through the global mobile network. NB-IoT was designed for low cost, extended coverage, reduced power consumption, and simplified device implementation. NB-IoT devices such as pet and vehicle trackers are already available on the market. Access authentication in NB-IoT devices is based on the traditional 3GPP mutual authentication scheme which relies on a shared-secret between the Subscriber Identity Module (SIM) in the device and the mobile operator. While some devices may require users to purchase a SIM card, many of them are already shipped with a SIM card and users can purchase various monthly subscription plans online. Thus, enabling Internet access on such devices requires minimal to no user interaction.

Extensible authentication protocol

It is obvious from the concise overview above that different protocols and credentials are used for network access authentication of IoT devices. We will now look at Extensible Authentication Protocol (EAP) [5] standardized by the Internet Engineering Task Force (IETF). Particularly, we will look at some features of the EAP framework which may make it suitable for network access authentication of IoT devices.

EAP is a lock-step protocol and only supports a single packet in flight. Nonetheless, EAP can support multiple authentication methods and can run directly over the link-layer without IP connectivity. The base EAP specification only defines the packet structure for simple request, response, success, and failure messages. EAP itself does not support fragmentation but implements retransmission and duplicate packet detection. Individual EAP methods use the base specification to build an authentication mechanism that allows devices and networks to identify and authenticate each other before making any authorization decision.

The initial EAP specification from 1998 was intended for devices on a Point-to-Point Protocol (PPP) link to authenticate each other before communicating. However, the usage of EAP is no longer limited to PPP and it is used in various network settings. In EAP, the entity requiring authentication is termed as the EAP authenticator while the other end point is referred to as the EAP peer. The specification also allows the use of a backend authentication server with the authenticator simply behaving as a pass-through. This can be helpful as the backend server can support many different authentication methods and new methods can be added without requiring changes in the access network. The entity where EAP authentication terminates is referred to as the EAP server. Thus, the EAP server can be part of the authenticator or the backend server. EAP is often deployed together with a protocol for authentication, authorization, and accounting (AAA) such as RADIUS [6] and DIAMETER [7]. When EAP is used with AAA protocols, the authenticator always acts as a pass-through. In such deployments, the AAA server, EAP server, and backend authentication server refer to the same entity.

Three example deployments of EAP are shown in Figure 1. In the simplest case (Figure 1a), an access point providing Internet access is the EAP authenticator and a laptop device wishing to receive connectivity is the EAP peer. Figure 1b shows a scenario where a backend authentication server is used for several devices on the network. Note that this server is located on the access point itself. Figure 1c illustrates a more common deployment of EAP in enterprise Wi-Fi networks where the backend RADIUS/DIAMETER server is on different host.

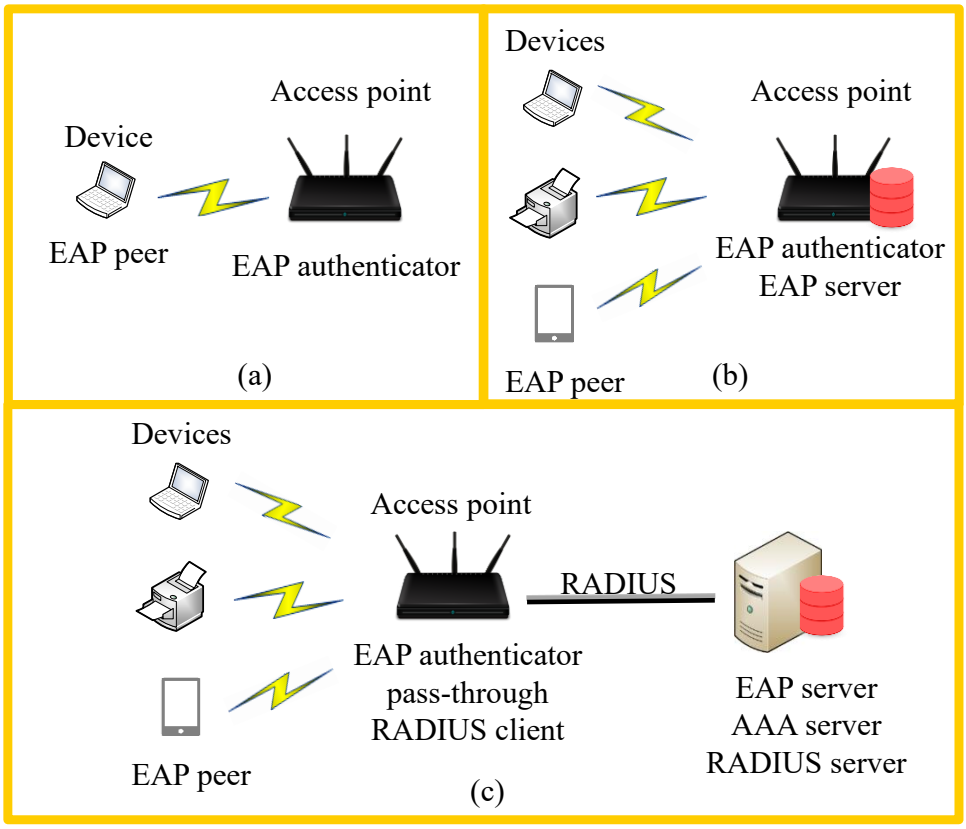

Figure 1: EAP deployment modes

EAP supports peer-to-peer operation where simultaneous authentication occurs in both directions. This implies that both sides have authenticator and peer functionality. This mode is however not commonly deployed, and it is instead more common to rely on EAP methods with mutual authentication.

An important aspect of EAP not discussed thus far is the identity used. EAP peers use a standardized identity called the Network Access Identifier (NAI). NAIs are utf-8 strings that are typically of the form: username@realm. The realm is a domain name which is used for routing EAP authentication to the correct server. For example, the access point in figure 1c can be connected to many different AAA servers and it can route EAP sessions based on the realm presented in an identity response message from the peer. The username part can identify a user but most EAP methods instead use anonymous identities. The real identity of the user is protected and only revealed to the server. In this article, we will discuss how the feature of routing EAP sessions to different backend AAA servers based on the realm can enable numerous deployments.

The discussion in the rest of this article may seem focused on network access authentication via an access point in Wi-Fi networks. However, the same principles are almost directly applicable to other IoT radio technologies. For example, the Zigbee IP [8] profile uses EAP for network access authentication. In other deployments, the access point maybe referred to as the coordinator (in Zigbee), the controller (in Z-wave [9]), or simply as a hub or gateway. Furthermore, the discussion in this article is intended for IoT devices with sufficient computational capabilities for performing access authentication with EAP (~32 kB of RAM and 512 kB of flash memory).

Many authentication methods

As noted in RFC 8576 [10], IoT devices are highly heterogeneous in terms of their computational capabilities, energy constraints, and user interfaces. Moreover, these devices are deployed in locations with varying level of network infrastructure. Manufacturers of IoT devices will also differ in terms of their security practices, user support, trustworthiness, and business models. Thus, while some device manufactures may install certificates on their devices or provide companion smartphone applications, it may be unwise to assume the same practice is followed by all manufacturers.

EAP can be a suitable common framework for this heterogeneity. Depending on the type of credentials, it has many authentication methods as shown in table 1.

Credential Type	**EAP Method**	**Specification**
Client and server certificates	EAP-TLS	RFC 5216
Pre-shared keys	EAP-PSK	RFC 4764
One-time passwords	EAP-POTP	RFC 4793
User typed passwords	EAP-Pwd	RFC 8146
SIM card	EAP-AKA`	RFC 5448
Server authentication with certificates and client authentication with user typed password	EAP-TTLS	RFC 5281

Table 1: EAP methods for different types of credentials

As stated in table 1, EAP methods such as Tunneled Transport Layer Security (EAP-TTLS) allow access authentication based on a combination of different credentials. Hence, one can use manufacturer issued device certificates along with user typed passwords before allowing a device to join the network. New EAP methods where authentication is based on data communicated over an OOB channel are also being standardized [11]. If, however, a desired authentication method does not exist, new EAP methods can easily be registered.

Many EAP methods support mutual authentication which can prevent IoT devices from joining open or insecure networks controlled by an attacker. Mutual authentication can also allow devices to opportunistically attempt and connect with all networks in its vicinity. Such opportunistic connection attempts may be necessary since the devices may not have the necessary UI for the user to configure the correct network. It may initially seem that such opportunistic connection attempts can leak device or user identities. However, the EAP specification recommends identity protection and many methods initially use anonymous identities while revealing the real device or user identity only after authenticating the network.

EAP also allows automatic selection of the appropriate authentication mechanism based on the device identity and local policy. This can be useful in situations where devices have multiple credentials, or, where devices with different credentials (and privileges) exist in the same network.

Local vs. cloud authentication

The deployment models of EAP shown in figure 1 already hint at its flexibility. This is especially relevant for IoT devices since they will be used in networks ranging from small home to large

enterprises. A simple deployment that uses EAP for network access authentication of IoT devices in a home network is shown in figure 2a. This deployment requires the access point to implement the correct EAP authentication method. If there are devices with different credentials, the access point must implement all the necessary EAP authentication methods.

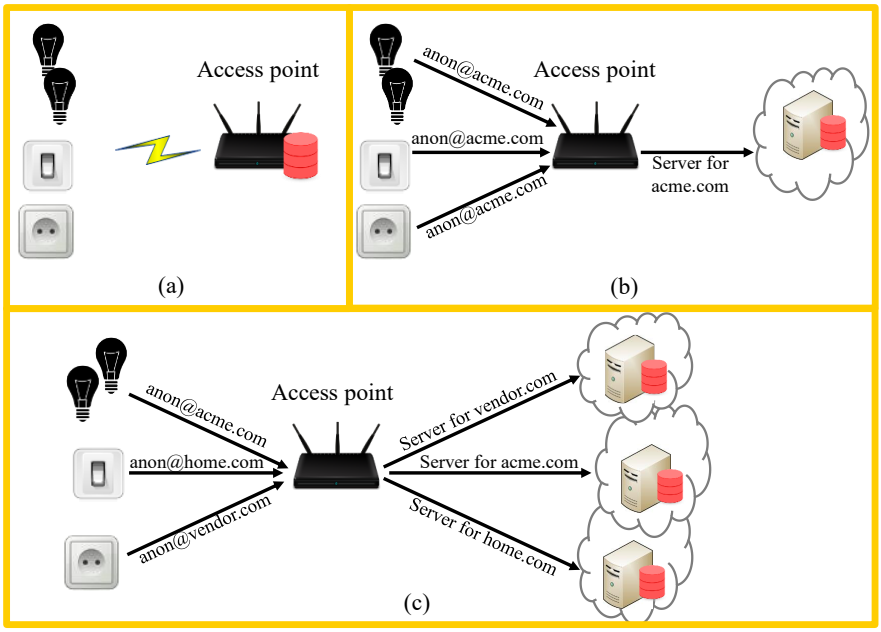

Figure 2: EAP deployments for IoT devices - Local vs. Cloud

The deployment model in figure 2a requires a local database of devices and credentials in the access point. The access point will also typically need a web server and corresponding UI for letting users monitor and manage their connected devices. This can be challenging because of the limited resources on the access point. Administering a local database in the access point can also be insecure since software updates for access points are rarely available or installed. Thankfully, the EAP architecture can alleviate such worries by removing the need for the user to administer a local database of client devices. Access authentication and management of devices can be delegated to a backend AAA server in the cloud as shown in figure 2b. Such a deployment scheme is not only easier to secure but also allows users to recover from situations where an access point is lost, reset, or merely upgraded to a newer version. The backend AAA server can also easily bind new devices to a specific user account in the cloud. Eventually, this backend AAA server for network access authentication may only be a small part of a holistic management server which provides services such as:

1. *Set up new devices*: Once the devices are securely associated with the user's personal account and have network connectivity, a user can instruct the device to connect to other services. A user can also rely on this management server to define permissions and access control lists.
2. *Software update*: The management server will receive device metadata (such as make and model) during the authentication process. This can enable the management server to keep track of the firmware and application software versions running on the device and inform the user about available updates. Since the management server has a security association with the user devices (resulting from EAP authentication), it can also schedule the updates. The managed update process could ensure timely installation of security patches and also

enable staged updates so that, for example, not all printers or light bulbs are out of use at the same time.

3. *Monitoring device behavior*: There are concerns that some IoT devices could spy on the users or participate in other unsavory activity, such as large-scale denial-of-service attacks. The devices could become compromised after deployment, but also by vendors in the distribution chain. In an advanced persistent threat (APT), a compromised device could be used as a steppingstone for attacking other devices in the same network. To mitigate such threats, the management server could monitor and control the network connections to and from the IoT devices.

Figure 2b shows a deployment where all the devices belong to the same vendor. However, most deployments will contain devices from multiple vendors. So the obvious next question is who will own and run the cloud service in multi-vendor deployments. Here again, EAP is flexible enough to route the authentication session to the backend servers of each vendor as shown in figure 2c. Alternatively, it is possible that all the devices are managed from a single cloud by forwarding all EAP authentication sessions to the same server irrespective of the device NAI. It is possible that this cloud-based management server will be provided by mobile network operators as a standard component of the telecom infrastructure. Cloud service providers could also host the management server for a fee, which may be subsidized with advertising and premium features. Finally, gateway or hub devices (such as SmartThings from Samsung) could also provide the service to home users, in which case they can preconfigure the gateway to connect to their server.

More advanced networks such as those in enterprises are managed by trained professionals and can maintain a local database of client devices. It is also possible to only maintain a server for traditional enterprise devices locally while outsourcing IoT devices to a remote server in the cloud.

Roaming and federation

Most IoT devices once deployed will likely remain static and not join new networks at different locations until they are decommissioned and transferred to a new owner. For example, smart light bulbs in a home are likely to remain static until the end of their functional lifetime. Nonetheless, there is niche category of IoT devices that will be mobile across networks and geographies. Devices such as pet trackers and connected cars will need extended coverage and service from many network providers.

Naturally, roaming requires business agreements between participating network providers. Once an agreement is ready, the EAP architecture already has all the components for roaming and accounting with RADIUS/DIAMETER readily available. Roaming in EAP is possible because the visiting network operator can use the realm part of the NAI to route the authentication. For example, a mobile pet tracker using LoRaWAN can receive Internet access to report the current location from a visited network as shown in figure 3.

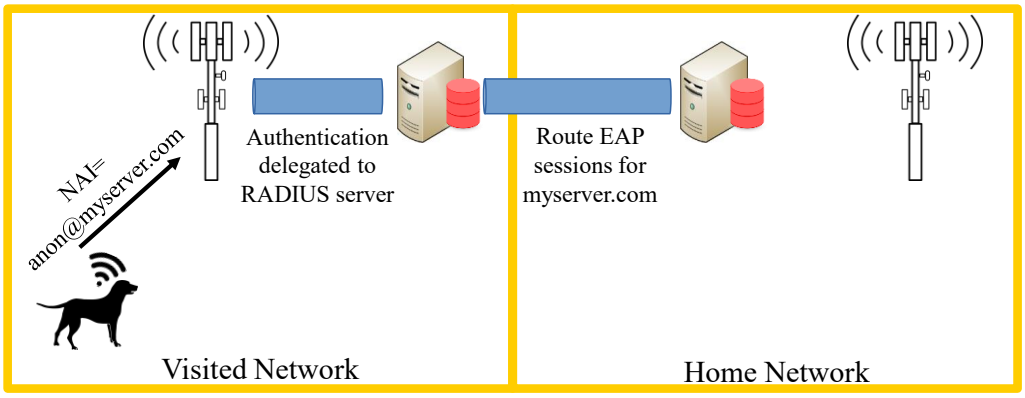

Figure 3: EAP roaming for IoT pet tracker

Lastly, the same roaming principles can also be applied for cross-organization access authentication. Eduroam [12] is an example of a federation that allows users from participating educational institutions to obtain Internet connectivity when visiting federation members. Eduroam also relies on EAP together with RADIUS for allowing such cross-organization network access. It is foreseeable that in future, eduroam will also allow users to connect their IoT devices. An exchange student for instance can use such a federation to obtain Internet connectivity for a smart plug while visiting a partner university.

Mergers, acquisitions, and manufacturer obsolescence

Thus far, we have discussed several deployment models that rely on the manufacturer of the IoT device for network access authentication. This is reasonable since most of the services offered by IoT devices are tightly integrated to- and provided by- the device manufacturer. However, it is not uncommon for startups to build IoT devices which are later acquired by well-established large enterprises. Thankfully, handling such mergers and acquisitions is routine in enterprise networks which rely on EAP for access authentication. The same flexibility can be extended to IoT devices if they rely on the EAP architecture with a backend AAA server.

There is also a concern among some that over-reliance on device manufacturers for initial network authentication is not always prudent. After all, not all device manufacturers are willing or capable of maintaining secure online services throughout the functional lifetime of a device. Manufacturers can stop supporting their IoT devices for a variety of planned and unplanned reasons. Nevertheless, users might want the ability to connect and use their devices even in the absence of manufacturer support. An obvious question arises: who would offer a service for devices which are no longer supported by the original manufacturer. Whether this is something that can be enforced by regulation or by business incentives is beyond the scope of this article. We do however note that there are examples of open-source communities supporting existing devices irrespective of manufacturer support: OpenWrt for home routers and Cyanogenmod (continued as Lineage OS) for smartphones. As discussed above, EAP allows the user to choose a backend server which does not belong to the original device manufacturer. It does require updating the forwarding rules based on the NAI. For most EAP methods, a manufacturer will also need to handover the credentials to the new backend server. However, authentication methods such as EAP-NOOB [11] do not require any pre-configured credentials and instead rely on opportunistic Diffie-Hellman key exchange with OOB authentication. EAP-NOOB allows users to readily migrate to any backend service of their choice without involving the device manufacturer.

Device isolation and revocation

Typical EAP deployments require unique credentials for each device instead of a network-wide passphrase. Such per-device credentials are also necessary in IoT networks to prevent one insecure IoT device from compromising the security of other devices on the same wireless network. Using EAP for access authentication can provide a tried-and-tested path to deployments with per-device credentials.

Another negative consequence of using the same network-wide passphrase is the challenge of access revocation for lost devices. Users are forced to refresh the shared passphrase and reconfigure all the remaining devices with the new passphrase. At first, this may seem acceptable in small home network deployments. However, as more devices such as locks, toys, light bulbs and other gadgets are added to home networks, network-wide shared secrets will no longer be sustainable. With the AAA server model discussed above, users can simply revoke access rights on the server for devices which are lost or sold.

Finally, some advanced enterprise Wi-Fi networks use EAP authentication to isolate devices into different virtual LANs (VLANs) with VLAN tagging. The AAA server sends a VLAN identifier to access points for successfully authenticated devices. Access points add the appropriate tag to the traffic and prevent hosts on different VLANs from communicating with each other without additional services. Such isolation is almost non-existent in most current IoT deployments. Some users are forced to use rudimentary mechanisms such as creating a separate SSID for IoT devices on the network. However, the need for such isolation is slowly being recognized. For example, the latest version of the Z-Wave [9] S2 protocol divides the network into dedicated security classes “S2 Access Control”, “S2 Authenticated”, and “S2 Unauthenticated”; each of which have a separate network key.

Issuing credentials with EAP

In some deployments, issuing new credentials to devices after EAP authentication may be necessary. For instance, a device may be shipped with a manufacturer certificate that cannot easily be revoked, replaced, or renewed. Some network operators in such scenarios will want to issue new credentials to devices after the initial authentication. These newly issued credentials can allow a device to access the local network and other services in the deployed network. It is however important to ensure that the newly issued credentials are securely bound to the initial credentials used for authentication. Tunnel Extensible Authentication Protocol (EAP-TEAP) already provides a mechanism for devices to request and receive certificates after authentication. Since EAP methods can easily be extended, the IETF is also considering new work on a generic mechanism for issuing operational credentials to IoT devices after the initial EAP authentication.

Implementations

EAP specification was completed more than a decade ago and no major errata or security vulnerabilities have been reported. It can safely be considered stable. The specification is also available without requiring any license and fee. In addition to the base specification, RFC 4137 [13] defines the state machines for EAP peers and authenticators. This ensures that implementations of EAP are highly interoperable not only in terms of the message structure but also in terms of the message processing and handling. Given that some IoT devices will be expected to remain

operational for many years, relying on a stable and mature standard can be beneficial for device manufacturers.

Moreover, several open source implementations of EAP are available and all major operating systems today ship with an EAP implementation. The lock-step nature of the protocol implies that EAP can be implemented easily even on resource-constrained devices as shown by Peltonen *et al.* [14]. Peltonen *et al.* also demonstrate that the overhead introduced by EAP packet headers is rather small when compared to the size of public keys and signatures. It is not surprising that several embedded operating systems such as ARM mbed and Amazon FreeRTOS already include support for some EAP authentication methods.

Encapsulation inside other protocols

EAP can be encapsulated inside various lower-layer protocols depending on the deployment needs. For example, EAP can be sent directly over the link layer without IP connectivity. IEEE 802.1x defines the encapsulation of EAP in IEEE 802 frames. Protocol for Carrying Authentication for Network Access (PANA) defines how EAP authentication can be transported using UDP after IP connectivity is obtained. PANA was designed for a network access authentication method that is agnostic of the link-layer. Encapsulating EAP inside RADIUS/DIAMETER is also standard practice in enterprise Wi-Fi networks. Due to the simple request/response nature of EAP, it can easily be encapsulated in many protocols. There is even a proposal to encapsulate EAP inside the Constrained Application Protocol (CoAP) [15] for IoT networks.

EAP in 5G

EAP is widely used for network access authentication in enterprise Wi-Fi networks. Initial versions of 3GPP specifications used EAP authentication only for integrated Wi-Fi networks that allowed operators to offload devices for enhanced capacity and coverage. Network access authentication in the core network relied on the Authentication and Key Agreement (AKA) protocol (called UMTS AKA and later EPS-AKA). With the introduction of 5G, there is a notable change. The core network now allows the use of EAP-AKA` for access authentication. 5G also specifies non-public networks (NPN) for enterprise and industrial deployments such as factories and campus networks. Network access authentication in standalone NPN networks (also called as private 5G networks) can be based on any EAP authentication method. For private 5G networks that are deployed in conjunction with a public mobile network, primary access authentication uses EAP-AKA` but secondary authentication can use any EAP method. It is evident that 3GPP realized the necessity for supporting different access authentication methods for the various target applications of 5G. EAP, with its support for many different credentials became an obvious choice.

Summary and remaining challenges

It is important to remember that many challenges with EAP remain. Users, for example, are accustomed to entering the same network-wide passphrase on all their devices. Transitioning to per-device credentials may require new tools and user training. Similarly, setting up AAA servers or RADIUS peering is currently not intuitive or user-friendly. New methods that hide these complexities are needed, particularly for small office and home networks. One such technology is network function virtualization (NFV), which can reduce the cost of spawning these security services.

Due to the heterogeneous nature of IoT, various standards will continue to rely on their own network access authentication method best suited for their devices and deployments. However,

encoding individual protocols in the EAP framework can provide several benefits as highlighted in this article. Moving to EAP can allow for future extensions and changes in deployment models.

References:

[1] Wi-Fi Alliance, "Device Provisioning Protocol Specification," ver. 1.0, 2018; https://www.wi-fi.org/downloads-registered-guest/Device_Provisioning_Protocol_Specification_v1.0.pdf/, accessed Dec. 28, 2020.

[2] ZigBee Alliance, "ZigBee Smart Energy Standard," rev. 19, ver. 1.2a, Dec. 2014; https://zigbeealliance.org/wp-content/uploads/2019/11/docs-07-5356-19-0zse-zigbee-smart-energy-profile-specification.pdf, accessed Dec. 28, 2020.

[3] Silicon Labs, "AN1089: Using Installation Codes with Zigbee Devices," rev. 0.5; https://www.silabs.com/documents/public/application-notes/an1089-using-installation-codes-with-zigbee-devices.pdf, accessed Dec. 28, 2020.

[4] LoRa Alliance, "LoRaWAN 1.1 Specification,"; https://lora-alliance.org/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf, accessed Dec. 28, 2020.

[5] B. Aboba, L.J. Blunk, J. R. Vollbrecht, J. Carlson, and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[6] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[7] V. Fajardo, J. Arkko, J. Loughney, G. Zorn, "Diameter Base Protocol", RFC 6733, Oct. 2012.

[8] Zigbee Alliance, "ZigBee IP Specification" Feb. 2013.

[9] ITU. 9959: Short range narrow-band digital radio communication transceiver-PHY and MAC layer specifications, 2012.

[10] O. Garcia-Morchon, S.Kumar, and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, Apr. 2019.

[11] T. Aura, M. Sethi, and A. Peltonen, "Nimble out-of-band authentication for EAP (EAP-NOOB)", Internet Draft, IETF, Dec. 2020.

[12] Eduroam, https://eduroam.org/, accessed Dec. 28, 2020.

[13] J. Vollbrecht, P. Eronen, N. Petroni, and Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", RFC 4137, Aug. 2005.

[14] A. Peltonen, E. Inglés, S. Latvala, D. Garcia-Carrillo, M. Sethi, and T. Aura, "Enterprise Security for the Internet of Things (IoT): Lightweight Bootstrapping with EAP-NOOB," *MDPI Sensors*, vol. 20, no. 21, 2020.

[15] D. Garcia-Carrillo, R. Marin-Lopez, "Lightweight CoAP-based bootstrapping service for the internet of things," *MDPI Sensors*, vol. 16, no. 3, 2016.

Biographies:

MOHIT SETHI (mohit.sethi@aalto.fi) works as a senior researcher at Ericsson and as a research fellow at Aalto University. He has received a Doctor of Science (DSc.) degree in Computer Science from Aalto University. He has previously completed his dual MSc. degree in security and mobile computing from Royal Institute of Technology (KTH), Sweden and Aalto University, Finland. Mohit actively contributes to IoT standardization at the IETF (author of RFC8387, RFC8576, and RFC8928). He is also currently chairing the Light-Weight Implementation Guidance (LWIG). Security dispatch (Secdispatch) and EAP Method Update (EMU) working groups of the IETF. He has received best paper awards at the ACM Ubicomp and IEEE IoT conferences. He has more than 50 international patent applications.

TUOMAS AURA (tuomas.aura@aalto.fi) received his MSc. and DSc. degrees from Helsinki University of Technology, Espoo, Finland, in 1996 and 2000, respectively. His doctoral thesis was on authorization and availability in distributed systems. He is a professor of computer science and engineering at Aalto University, Finland. Before joining Aalto University, he worked with Microsoft Research, Cambridge, UK. He is interested in network and computer security and the security analysis of new technologies. In addition to academic research, he works on industrial applications and standardization.