January 28, 2019

ZERO-TOUCH BOOTSTRAP OF A NETWORK CONNECTED DEVICE

Owen Friel

Eliot Lear

Jerome Henry

Stephen Orr

ZERO-TOUCH BOOTSTRAP OF A NETWORK CONNECTED DEVICE

AUTHORS:
Owen Friel
Eliot Lear
Jerome Henry
Stephen Orr

ABSTRACT

Techniques are presented herein that allow devices to automatically discover the correct enterprise wireless network to connect to, and securely onboard against that network, without manual provisioning of network information or credentials on the devices. This enables secure deployment of devices at scale on enterprise wireless networks. Minor enhancements to Institute of Electrical and Electronics Engineers (IEEE) 802.11 are described to enable this flow. Unlike Wi-Fi Alliance® Device Provisioning Protocol (DPP), the techniques presented herein are lightweight and do not include additional messaging overhead between the client device (STA) and Access Point (AP).

DETAILED DESCRIPTION

When network connected devices (e.g., Internet of Things (IoT) devices/Things, Collaboration endpoints, etc.) are powered up for the first time, initial onboarding against secure wireless networks enforcing authentication (e.g., Institute of Electrical and Electronics Engineers (IEEE) 802.1x authentication) is generally a laborious manual process. In particular, in the case of 802.1x, a Service Set Identifier (SSID) string and an authentication credential must be provisioned on the device. This is a significant "Total Cost of Ownership" and "User Experience" issue for even a small number of devices, and simply does not scale for large numbers of devices.

The newly published Wi-Fi Alliance Device Provisioning Protocol (DPP) specification attempts to address this problem by defining how a Configurator application can enable a device to discover and connect to a Wi-Fi® SSID. In particular, the Wi-Fi Alliance DPP allows for the Configurator application to pass two main categories of credentials to the device, namely:

1. A password or passphrase which can be used to connect to a wireless network that is enforcing a password based Authentication and Key Management (AKM) of

either Simultaneous Authentication of Equals (SAE) or Pre-Shared Key (PSK), and advertising this in the Robust Security Network Element (RSNE) Beacon and Probe Responses; or

2. A DPP Connector object which can only be used to connect to a wireless network that is enforcing the newly defined "dpp" Authentication and Key Management (AKM), which is advertised via a Wi-Fi Alliance Organizationally Unique Identifier (OUI) in the RSNE.

Neither of these approaches are suitable for connection to enterprise networks enforcing 802.1X (and therefore rejecting the use of PSK, SAE or a particular "dpp" AKM). The techniques presented herein solve the problem of IoT dynamic provisioning in the context of an enterprise environment.

More specifically, the techniques presented herein define an enhancement to 802.11 Authentication frames that enables client devices (STAs) to onboard securely against an 802.11 network that is enforcing 802.1X authentication while providing proof to the STA that the network has knowledge of the STA's IDevID public key. Additionally, the techniques presented herein protect disclosure of the STA's IDevID public key to rogue 802.11 networks.

In accordance with the techniques presented herein, an 802.11 Proof of Knowledge (POK) Authentication Algorithm is defined that is used by the STA to verify that an Access Point (AP) knows the STA's IDevID public key prior to attempting to connect to the AP. This also enables the device to protect its IDevID public key from exposure to rogue networks.

In addition, the techniques presented herein also describe multiple potential mechanisms for how Bootstrapping Remote Secure Key Infrastructures (BRSKI) may be integrated into the device onboarding flow. The techniques presented herein do not rely on DPP, and do not use any of the newly defined messages in DPP. Moreover, the techniques presented herein do not rely on the presence or use of a helper application on a mobile device or tablet in order to bootstrap the STA, and may be fully integrated into the AP. The network enforces an Extensible Authentication Protocol (EAP) Transport Layer Security (TLS) method such as EAP-TLS, EAP Tunnel Extensible Authentication Protocol (EAP-TEAP) or any EAP method that establishes a TLS connection.

Figures 1A-1C below collectively illustrate a call flow in accordance with embodiments presented herein. Details of the call flow shown in Figures 1A-1C are described further below.

FIG. 1A-802.11 Proof of Knowledge (POK) + BRSKI (part 1)

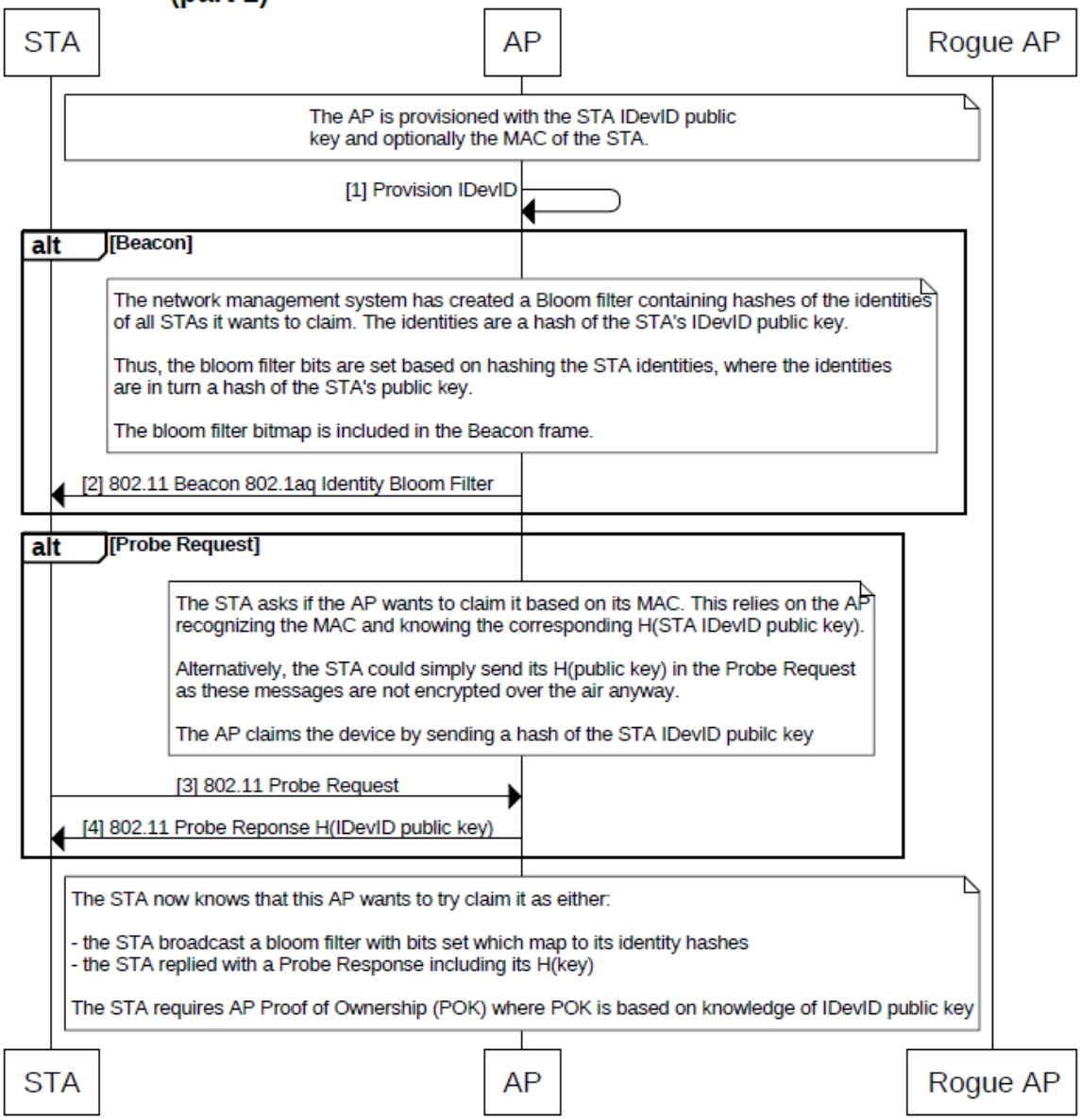


FIG. 1B- **802.11 Proof of Knowledge (POK) + BRSKI (part 2)**

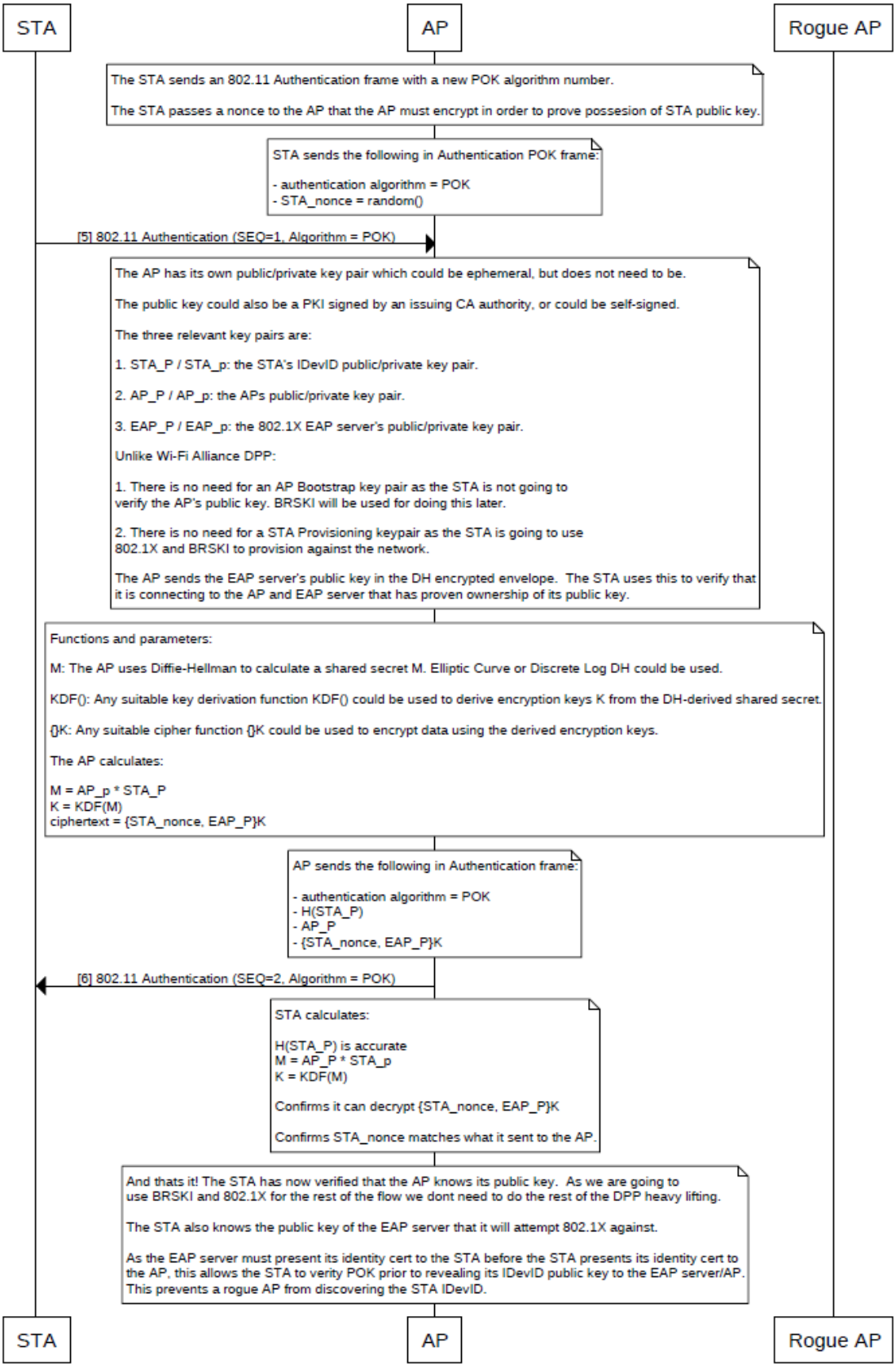
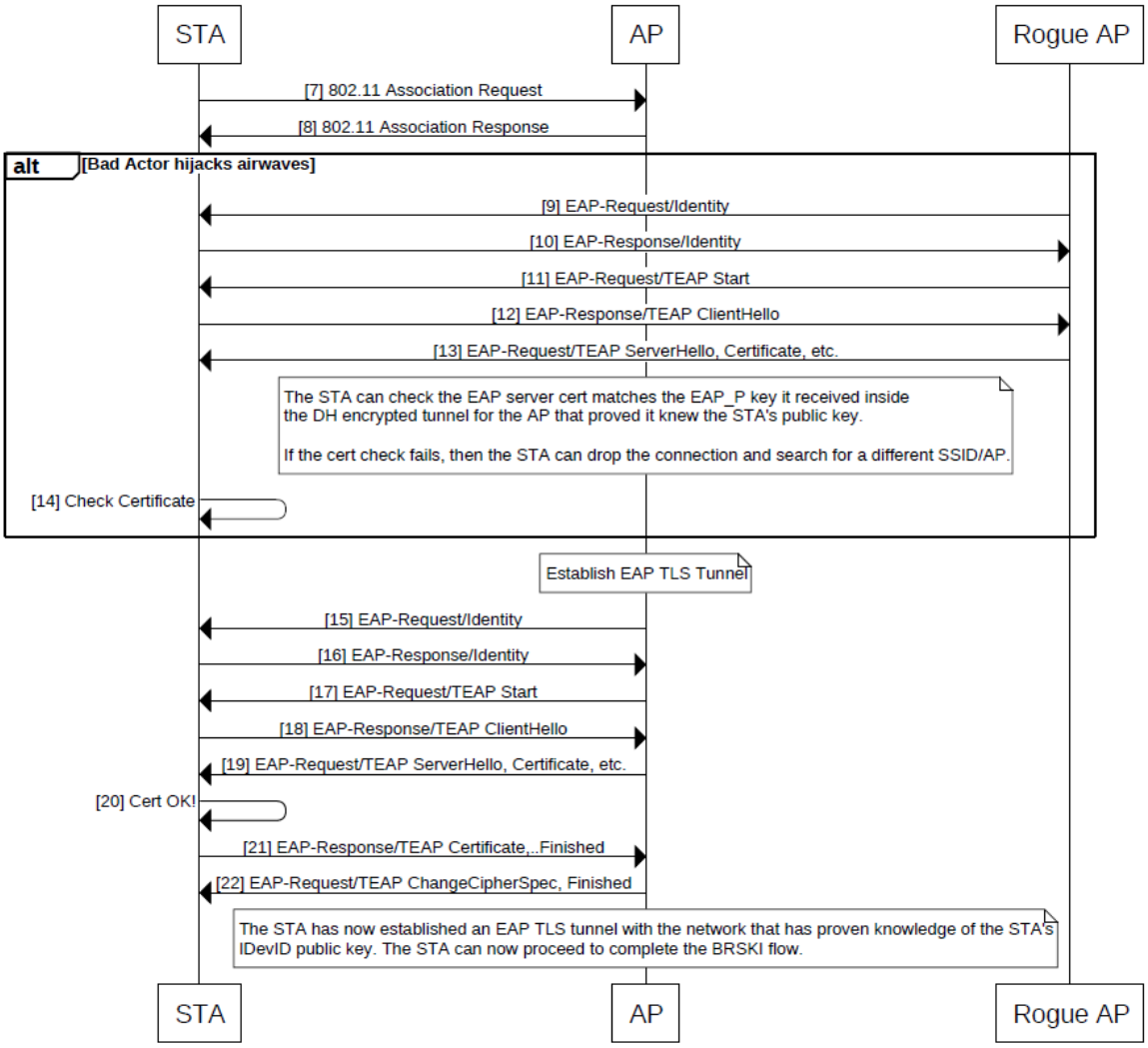


FIG. 1C- **802.11 Proof of Knowledge (POK) + BRSKI (part 3)**



In Step 1 of the call flow, the administrator (admin) provisions the STA IDevID public key on the network. The admin may optionally provision the STA Media Access Control (MAC) address on the network. This may be done in advance of the STA being powered up, for example, when the STA arrives in shipping. This step may be done in batch for a large number of expected STAs, and prevents unauthorized STAs from automatically joining the network.

Note that the STA IDevID and, optionally the MAC address, may be provisioned on any suitable element in the network. For example, they could be configured on the Authentication, Authorization and Accounting (AAA) server or on the Wireless Local Area Network (LAN) Controller (WLC) elements. In certain implementations, this phase can

be performed against authentication databases. The network element can then provide this information to the required set of APs when necessary.

Two embodiments for initial discovery of the AP are described below:

Embodiment 1:

In Step 2 of the call flow, the network management system has created a Bloom filter containing hashes of the identities of all STAs that it wants to claim. The identities are a hash of the STA's IDevID public key. Thus, the Bloom filter bits are set based on hashing the STA identities, where the identities are in turn a hash of the STA's public key. The Bloom filter bitmap is included in the Beacon frame.

If the STA sees a Bloom filter match then it knows that the AP is likely attempting to claim it.

Embodiment 2:

In this embodiment, the Beacon at step 2 of the call flow does not contain a specific identifier helping the STA realize that the AP is trying to claim it. In Step 3 of the call flow, the STA sends a Probe Request with a suitable flag set indicating that it wants to enroll using POK of its IDevID.

The “alt Probe Request” section of the call flow provides two options. In the first option, the AP must know the MAC address and map the MAC address to the STAs H(public key). In the second option, the STA simply sends its H(public key) to the AP. This avoids the overhead of the network having to be pre-provisioned with the MAC address of the STA, or a map of MAC address to public key. There is also no real risk in the STA sending its H(public key) to the wrong AP in step 3, as none of these packets are encrypted so any rogue AP could sniff the H(public key) in step 4.

If the AP recognizes the MAC address of the STA, in Step 4 of the call flow, the STA sends a Probe Response which includes a hash of the STA's IDevID public key. In an alternative embodiment where the AP does not contain all the IDevID public keys, the AP can reply with a 'hold' flag while it retrieves the public key matching the STA MAC address, and then send a gratuitous Probe response with the hash.

At Step 5 of the call flow, the STA knows that the AP wants to claim it as either:

(i) the AP advertised a Bloom filter with matching bits set, or

(ii) (ii) the AP replied with a Probe Response including a hash of the STA's public key

In Step 5, the STA sends an 802.11 Authentication message indicating the algorithm type of POK. The STA includes a random nonce in the request.

On receipt of the message, the AP derives a shared secret via a Diffie Hellman (DH) calculation over a finite field (and it could be discrete log or elliptic curve) using the AP's private key and the STA's public key. The AP uses the DH shared key and a suitable key derivation function to derive shared encryption keys.

It should be noted that although the IDevID public key is used for the initial exchange, the purpose is to create a tunnel with a proof that the infrastructure side has knowledge of the device keying material. As such, alternate options (e.g., pre-set PSK, other form of public key exchanges, or a suitable Password Authenticated Key Exchange (PAKE) algorithm (e.g., PAKE by Juggling (JPAKE) or Secure Remote Password (SRP)) may also be applicable.

The AP uses a suitable encryption algorithm to encrypt the following data:

- the STA's nonce received in the Authentication message

- the public key of the EAP server that will be enforcing 802.1x EAP TLS authentication

In Step 6 of the call flow, the AP replies with an 802.11 Authentication message indicating the algorithm type of POK. The AP includes its DH public key, the encrypted data, and the hash of the STA's public key.

On receipt of the message, the STA derives the same DH shared secret and encryption keys, and verifies that it can decrypt the ciphertext. The STA verifies that the nonce received from the AP matches the nonce it originally sent. At this point, the STA has now verified that the AP knows its public key, and the STA proceeds to connect to the AP and initiate the 802.1X authentication. Steps 7 and 8 of the call flow generally illustrate a standard 802.11 Association exchange.

The STA has also received the public key of the EAP server for the network that has completed POK of its IDevID public key. At this stage there is still no secure encrypted channel between the STA and the AP. As such, a malicious attacker could still try to hijack the connection. However, when establishing the 802.1X EAP TLS tunnel, the STA uses

the received EAP server public key to confirm that it is doing the 802.1X handshake with the correct network.

Steps 8 through 14 of the call flow illustrate how the STA uses this to prevent a rogue network from hijacking the connection. When the rogue network presents its server Certificate in Step 13 of the call flow, the STA compares it with the key received inside the POK DH envelop, and drops the connection if the keys do not match.

Steps 15 through 22 of the call flow illustrate a successful EAP authentication flow. When the network presents its server Certificate in Step 19, the STA compares it with the key received inside the POK DH envelop, and it confirms a match and completes the 802.1X TLS handshake in steps 21 and 22 of the call flow.

At this stage, there are multiple potential integration options possible for proceeding with a BRSKI flow whereby the device uses its IDevID to establish trust with the local network, enroll and obtain an LDevID in accordance with techniques presented herein. These options include:

- Option 1: an EAP-BRSKI method could be defined and a tunneled EAP mechanism could be used to tunnel the inner EAP-BRSKI method inside the outer EAP TLS tunnel. The BRSKI flow would complete inside the context of the EAP authentication prior to the STA receiving its IP address.
- Option 2: EAP BRSKI Type- Length-Values (TLVs) could be defined and these TLVs transported inside an outer EAP-TEAP TLS tunnel. The BRSKI flow would complete inside the context of the EAP authentication prior to the STA receiving its IP address.
- Option 3: The EAP authentication completes and the BRSKI flow happens at the application layer after a Layer 3 connection is established and after the STA has received its IP address.

In each of the above options, once the BRSKI flow is complete and the STA has received the local domain trust anchors, the STA substantially immediately tears down the connection if it fails to verify the EAP server's identity using those anchors.

Once BRSKI is complete, the device can enroll to obtain its LDevID, and can use its LDevID for all subsequent network accesses and 802.1X authentication.

In one example, presented is a method to integrate a mutual recognition and network parameter provisioning between an IoT object and an enterprise management network over Wi-Fi. POK may be exchanged over Wi-Fi. An AP may provide POK over stateless exchanges. The IoT device may automatically choose the correct Wi-Fi network to join. This process may be automated (e.g., no manual intervention on individual IoT devices is needed as they are installed and powered on).

As described herein, IoT devices may be installed individually by Operational Technology (OT) teams without the need for Information Technology (IT) team shadowing. Moreover, these devices may be turned on massively ('building scale'). That is, there is no device by device action. The IoT devices may automatically pick the right SSID (e.g., in multi-tenant buildings with multiple IoT-friendly SSIDs). Further, IoT devices may be provided assurance that the picked network is legitimate (e.g., 'my' network, not 'neighbor' network or 'rogue' network), regardless of the underlying 802.11 authentication method (e.g., works in Open, Pre-Shared Key (PSK), or 802.1X/EAP Wi-Fi networks). The recognition choreography described herein allows for integration into a standard 802.11 network while producing the aforementioned results.

Provided is a method by which the probe request can result in a comeback exchange. By using the comeback mechanism, the AP may collect the information it needs to answer a particular question, without mandating the AP to convert the probe exchange to a stateful exchange. This structure may be used to allow for many other applications.

Information Element (IE) bit values may be used which represent "special meaning." Because the exchanges are integrated into the standard 802.11 choreography, the method described may be integrated into any 802.11 driver or network. Therefore, knowledge of the devices (e.g., with BRSKI, IDevID, etc.) may be first installed into the infrastructure at any point prior to the IoT device being added to the network. Once the IoT device is added to the network, no manual IT intervention is needed. This allows the IoT device operator to be an OT person, without IT shadowing.

The same method may be applied to radio technologies other than 802.11 (e.g., Bulk Transmission (BT) and other 802.15.4 techniques), because the existing discovery chorography and messages may be reused.

This method may be independent of the underlying network authentication method. For example, in the case of 802.11, this method may be implemented if the production network is Open, PSK or based on 802.1X/EAP authentication.

There are several possible touchpoints for an IoT onboarding process. The first possible touchpoint is network infrastructure (e.g., AP, WLC, etc.). This is simply about network configuration (i.e., a "configure once" type of action). The second possible touchpoint is an authentication database, and the third possible touchpoint is a further management platform (e.g., mesh application and service architecture). The second and third possible touchpoints may be mass-configured (upload of batches). Therefore, although there is a need for configuration for these items, they represent a low level of effort.

The fourth possible touchpoint is individual IoT devices, one at a time. This functions at a different scale from the other possible touchpoints, and this is the problem addressed herein. When each individual device needs to be configured, the required effort is linear with the number of device. This task is insignificant at small scales (e.g., a house), and can sometimes be integrated in the deployment phase. However, at large scales the manual configuration and onboarding this possible touchpoint does not work anymore. For example, when an entire building is retrofitted, and then power is turned on, this can cause thousands of IoT devices to come online at the same time. Alternatively, if the OT and IT teams coordinate, this can result in unpredictable (from the IT standpoint) onboarding needs and cycles.

As such, techniques described herein allow IoT devices to automatically onboard and join a network without the requirement for operators to configure or manually interact with each individual IoT object.

802.11 is not modified in a fundamental way (i.e., no modification of a protocol). 802.11 allows some frames to carry specific Information Elements, which value can be set to known meanings, or can be set to a value that has a 'special meaning'. The special meanings may be defined to support the IoT onboarding use case. As such, 802.11 is not being modified, but rather a meaning is being specified for certain frames allowed by 802.11. The possibilities offered by the probe exchange, are thereby extended by allowing additional information to be exchanged while still keeping the fundamental stateless nature

of the exchange. The method described herein may be integrated into any existing 802.11 stack with only minor modifications.

In summary, techniques are presented herein that allow devices to automatically discover the correct enterprise wireless network to connect to, and securely onboard against that network, without manual provisioning of network information or credentials on the devices. This enables secure deployment of devices at scale on enterprise wireless networks. Minor enhancements to IEEE 802.11 are described to enable this flow. Unlike Wi-Fi Alliance DPP, the techniques presented herein are lightweight and do not include additional messaging overhead between the STA and AP.