DOI:10.1145/3637830

Article development led by acmqueue
queue.acm.org

FDO's untrusted model is contrasted with Wi-Fi Easy Connect to illustrate the advantages of each mechanism.

BY GEOFFREY H. COOPER

Device Onboarding Using FDO and the Untrusted Installer Model

THE INTERNET OF Things (IoT) market has grown into a major business, with millions of devices and servers dedicated to tracking, logging, and measuring real-world features. This covers many areas of society, including home, retail, manufacturing, street lighting, and transportation.

All the devices, in all these fields, share an important characteristic: They were manufactured under some initial ownership, and they were transferred into their target application, coming under another ownership. Only in the target context does the IoT device perform its intended function while interacting with supporting servers. The challenge is to set up this interaction in a manner that is fast, reliable, and secure. This process is referred to as *onboarding*.

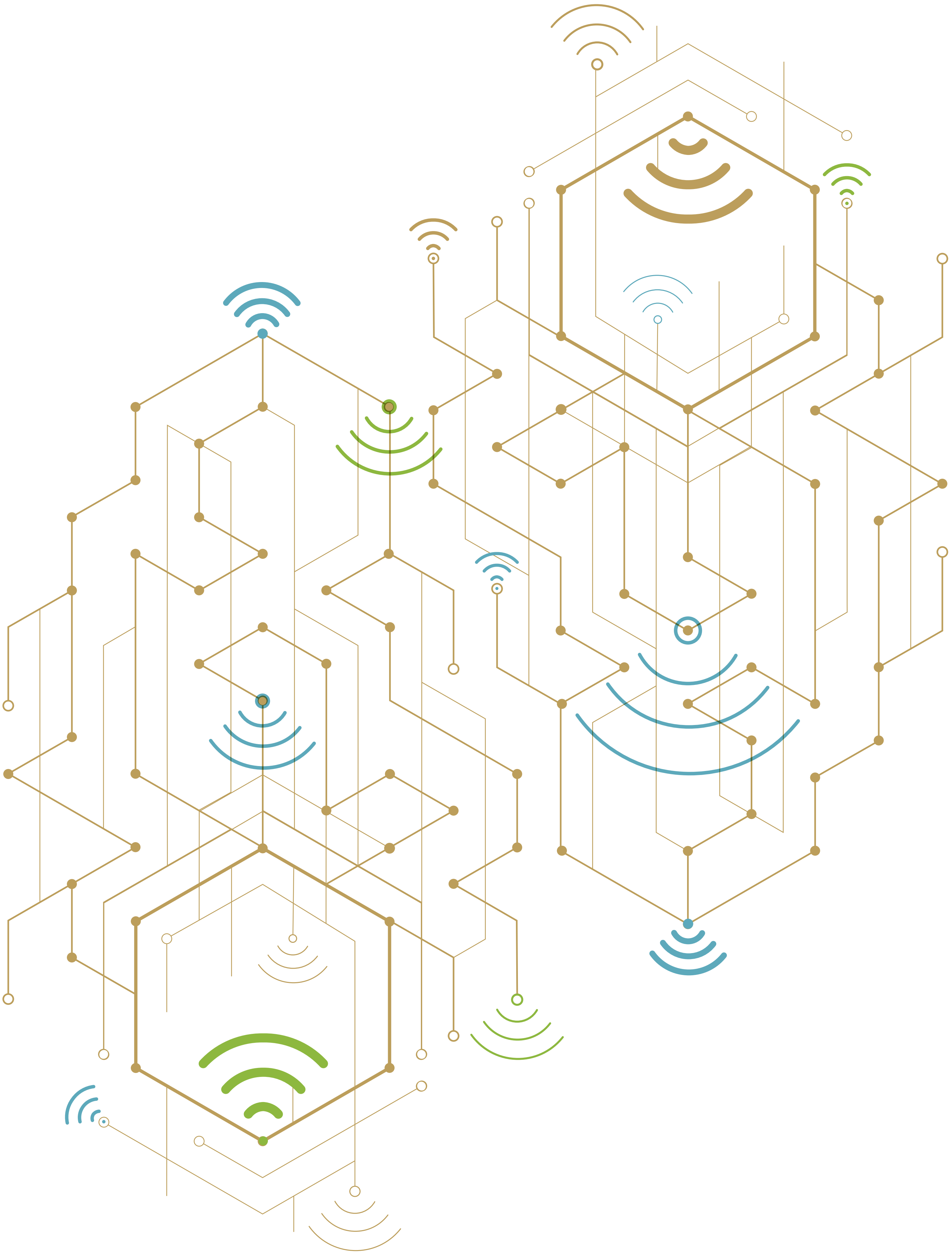
Table 1 outlines a simplified onboarding terminology for this article. Both machine and human entities are involved. Onboarding involves creating trust between a target *device* and a target *server*, with enough mechanisms that the server may subsequently communicate with the device and control some or all its functions. The *installer* is a person, trusted to install the physical device in a particular place; report which device is installed where; and connect the device to power and, if necessary, network cabling or wireless networking. In some cases, the installer interacts with programs on the device and server using tools that are collectively called the *configurator*.

The simplest way to onboard an IoT device is for an installer (person) to reset the device to factory settings, then install, by hand, all the software and credentials needed for the application to access the target service. When the installer hits the final Enter key, the device is onboarded and in service.

This has been the default model for onboarding devices in legacy applications, but it scales poorly. Problems that are likely to occur include:

- Incorrectly installed software, or wrong version used.
- Incorrect or missing credentials for the device.
- Inconsistent installation across multiple devices.
- Device inadequately secured into its new environment.

The obvious answer to all these problems is to automate the onboarding process. Typically, this means running a script that installs all the correct software and credentials on the device, verifies the installation, then secures the device. To install and run the script requires the device to have at least some security disabled, so the script must also enable security.



But the device cannot be secured from the installer, who runs the onboarding script. The installer needs both privileged status (for example, *root* access) and visibility into all the credentials that are added to the device. At the instant that onboarding is complete, an installer, previously trusted, becomes the weak link for mistakes or attacks. If the installer is a contract employee of another company or organization, the very knowledge of these secrets outside the company security perimeter constitutes a built-in security breach.

The alternative is to automate onboarding, so the installer who keeps with the principle of "least privilege" has access to the device and server only for the purposes of physical installation and initiating the automatic onboarding process. The installer never learns enough to breach the security relationship created during onboarding.

Various mechanisms are used to implement least-privilege access for the installer in automated onboarding, such as *trusted installer* and *untrusted installer* protocols. In a trusted installer protocol, the installer (a person) transfers specific information that allows the device and server to establish trust. After this, the device and server perform the rest of the onboarding protocol, so the installer does not learn the credentials derived by the onboarding process. In the untrusted installer protocol, the installer has no role in the onboarding communications protocol. The device and server must find each other using preprogrammed credentials and/or clues in the network environment.

An example of the trusted installer model is the Wi-Fi Easy Connect protocol from the Wi-Fi Alliance, which allows a device to connect to a local Wi-Fi router.[12] The installer runs a configurator app connected to the same router. The goal of the onboarding is network admission; the credentials passed for onboarding are relatively simple.

The FIDO Device Onboard (FDO) protocol is an example of an untrusted installer.[6] In FDO, an explicit credential, called the ownership voucher, is issued for each device. This credential is installed into the server to allow the device to onboard. FDO is an example of application onboarding, so it has a rich set of primitives to allow credentials and other onboarding materials to be configured.

This article looks at each of these protocols to contrast the trusted and untrusted installer models for onboarding.

Basic Cryptography for Onboarding

Onboarding protocols use cryptography. Although this is not a discussion about cryptographic methods for onboarding, a basic outline of the techniques is necessary.

The details of the cryptography used for the two protocols under discussion vary, but the basic components fit into the following model.

First, the server and device must authenticate, so that each has confirmed the trustworthiness of the other to onboard. In some protocols, server authentication is based on the environment and is considered optional.

Client (device) authentication is always mandatory. Authentication usually uses a digital signature. In some cases, the public key is wrapped in a digital certificate, usually in X.509 format, which implies public-key infrastructure (PKI).[5]

Since certificates have an expiration date, and since devices in the supply chain may sit on a shelf for months before being installed, conventional PKI methodology presents challenges for automatic onboarding. Sometimes expiration dates are extended for this purpose. Alternatively, public-key cryptography is deployed without certificates, so that the public key's cryptographic material is surrounded only by protocol fields.

After authentication, a cryptographic key exchange is performed.[1] A popular method is Elliptical Curve Diffie-Hellman (ECDH). Since ECDH is subject to a so-called manipulator-in-the-middle (MITM) attack,[11] a digital signature is also needed in this context. The authentication and key-exchange elements are sometimes combined in the protocol so that one digital signature can cover both authentication and key exchange, while also optimizing the number of message round trips. This makes it more difficult to map the protocol messages to the operations.

Key exchange establishes a shared secret known only to the device and server. A key derivation function (KDF)[3] is used to generate common credentials and establish an authenticated, encrypted tunnel between server and device. Then onboarding information may be shared over the tunnel, visible only to the device and server at each "end" of the tunnel.

Wi-Fi Easy Connect

Wi-Fi Easy Connect is a feature of Wi-Fi Protected Access 3 (WPA3) from Wi-Fi Alliance. It provides automatic configuration of a device to access a Wi-Fi network using the trusted installer model. The installer uses a configurator program, which must already be connected to the target network.

The configurator may be an app on a cellphone or Wi-Fi gateway. It takes on the server role in onboarding, establishing trust with a device, and shares the credentials for the network with it. Afterward, the device can use these credentials to enter the network as needed.

The Wi-Fi Alliance specifications

Table 1. Onboarding terminology.

Term	What	Role
Device	Computer, for example, IOT computer.	Computer to be onboarded.
Server	Computer, for example, IOT gateway, datacenter computer.	Computer that will control some functions of the device after onboarding.
Onboarding	Establishing a trusted, controlling relationship between a device and a server.	Device allows the server to control some or all functions of the device.
Installer	Human being.	The person who has "hands on" the device for its physical attachment and power on.
Configurator	Tool for installer to use, may have software and hardware components.	A tool used as an agent of the installer to interact with some functions of the device during installation.

are complex, having been developed to satisfy many diverse communications requirements. Wi-Fi Easy Connect has many options and interacts with other Wi-Fi standards. This article describes a simplified single path through these standards and is not intended to represent the full power of the collective Wi-Fi standards. (See the Wi-Fi Easy Connect specification for more details.[12])

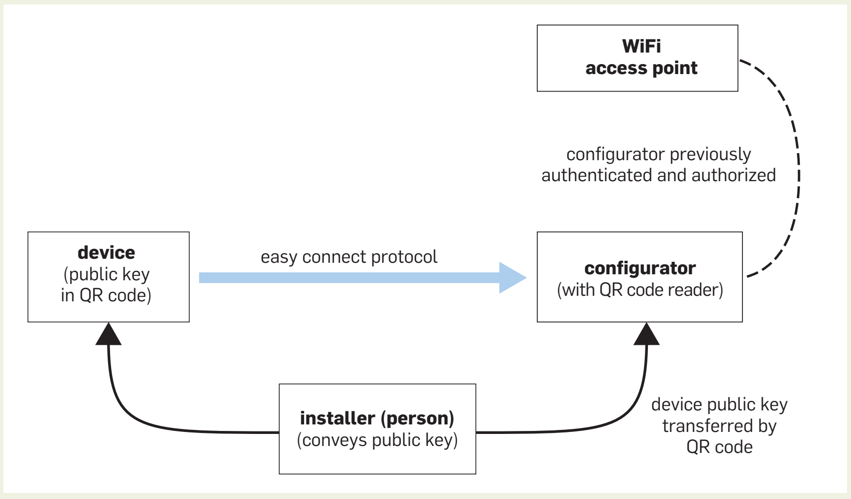
To support Wi-Fi Easy Connect, a device contains an identifying asymmetric key pair. The private key is embedded in the device and is accessible only to the device's software or firmware. The public key is imprinted on a QR code (a rectangular scanning code) that comes with the device, perhaps on a sticker affixed to the back. Other information is present in the QR code (for example, a list of frequency bands) but does not affect the integrity or security of onboarding. Wi-Fi Easy Connect can also use mechanisms other than a QR code.

An installer who wishes to attach the new device to the network uses the configurator to scan the device's QR code. This action identifies and authorizes onboarding the device to the configurator. The device (still powered off) can now authenticate itself using its private key. By default, there is no strong authentication of the configurator to the device. Wi-Fi WPA3 optionally adds bidirectional authentication (for example, for enterprise networks). Figure 1 illustrates the installation process with Wi-Fi Easy Connect.

Next, the installer powers on the device. Since the device is not yet connected to a Wi-Fi network, it sends a broadcast Wi-Fi *beacon* message to announce its availability. This allows the configurator to sense the device and connect to it using a one-to-one Wi-Fi connection called Wi-Fi Direct. If needed because of hardware limitations, the configurator may temporarily drop the connection to the target Wi-Fi network and talk only to the device.

Once a configurator has connected to it, the device initiates a key exchange protocol, signed with its private key. The configurator, having received the device's public key from the QR code scan, uses it to verify the signature and key exchange, and shares its own key-exchange information, allowing a mutual shared secret to be created. From this shared secret, keys for authentication and encryption are derived, sufficient to create a secured, authenticated, encrypted tunnel. An onboarding payload is transmitted across this tunnel, containing the long-lived parameters of the target Wi-Fi network. For example, a home network's service set identifier (SSID) and password might be transmitted.

Figure 1. Trusted installer.


WiFi access point
configurator previously authenticated and authorized
device (public key in QR code)
easy connect protocol
configurator (with QR code reader)
installer (person) (conveys public key)
device public key transferred by QR code

Once the onboarding payload is received and acknowledged, the onboarding operation is complete. The device and configurator terminate their private connection, and the configurator continues as a member of the target Wi-Fi network. The device is now also able to use its downloaded network parameters to connect to the target network.

Although the onboarding payload for Wi-Fi Easy Connect is small, this is not a limitation of the technique. Since it creates an authenticated tunnel, the trusted installer method can provision an arbitrarily large payload to the device.

The trusted installer has functioned as a key component of the onboarding, choosing the device to onboard and selecting the network and configurator program to interact with the device. Even so, the installer is not required to modify the device's software or firmware, or perform any significant operation on the device, other than to turn it on. Although the installer has arranged for credentials to be shared between the device and the configurator, neither device nor configurator needs to expose these credentials to the installer.

Indeed, if the installer can cause the configurator to scan a barcode, the installer might not actually be able to log in to the network at all. Trust in the installer is pared down to an introductory role only. For example, a home router might include a configurator program (and a barcode scanner), removing the need for the installer to access the local network. A plumber installing a home dishwasher might use the home gateway to scan the Easy Connect QR code and provide dishwasher access to the home network. The only network information that leaves with the installer is that the new dishwasher is connected to it.

On the other hand, an installer might not be trusted just by being in the home. The configurator must be secured by ad hoc mechanisms; otherwise, anyone in—or close to—the home could admit devices to the home network. For example, a visitor with configurator access could install and leave behind a camera that streams video to the Internet.

With the proviso that the configurator program is secured, the trusted installer provides a powerful and easy-to-implement technique for onboarding devices. It is often used for devices in home networks. In commercial or industrial settings, it is ap-

FDO Credentials

FDO is implemented by interaction among credentials that include:

- **A GUID that identifies the instance of credentials. It is used to pair the ownership voucher with the device.**
- **A device asymmetric key pair (private key and certificate), where the private and public keys uniquely identify the device.**
- **A set of instructions, called RendezvousInfo, to find a mutually agreed-upon rendezvous server. In the simplest case, this is the DNS name of a rendezvous server, whose operation is described in the article.**
- **An ownership voucher, a new cryptographic credential, defined for FDO. This voucher contains a ledger of public keys for supply-chain entities. The last key in the ledger is owned by the server that runs the FDO protocols to onboard the device. Note that, although the term ledger is commonly used for blockchain devices, the ownership voucher does not use blockchain technology.**
- **A private and public key pair for each supply-chain entity that processes the ownership voucher.**
- **Additional credentials, provided for practicality and convenience.**

propriate when the trust relationship to the installer is well understood. This may require, for example, that only direct employees take on the trusted installer role in a corporate network.

FIDO Device Onboard

FDO is published as a standard by the FIDO Alliance. It is the first technical publication of the FIDO Alliance's IoT Technical Working Group. FDO is distinct from the previous authentication-based standards published by FIDO, sometimes called FIDO2.[7] The description here is based on FDO version 1.1.[6]

FDO data formats are built on several Internet Engineering Task Force (IETF) standards. FDO messages and data objects (including the ownership voucher) are encoded using the Concise Binary Object Representation (CBOR) format.[2] Digital signatures use the CBOR Object Signing and Encryption (COSE) format[10] or the Entity Attestation Token (EAT).[8]

FDO is an automatic onboarding protocol based on the untrusted installer model. Since the installer is not trusted to interact in the protocol, there is no configurator role. The server and device have all the information needed to onboard the device. All the installer does is press the device's power button.

In FDO, two sets of related credentials are created when the device is initialized, usually during manufacturing. One set of credentials is placed in the device. The other credentials are used to create an object called the ownership voucher. It is cryptographically linked to the device's credentials, allowing the voucher to be used to authenticate the server. Figure 2 illustrates the installation process with FDO.

The device and the server are both identified by asymmetric key pairs. The device's private key is secreted in the device, and the public key of the device is in an X.509 certificate created by the device manufacturer, itself embedded inside the ownership voucher. The server is identified by an asymmetric key pair. The private key lives in the server, and the public key lives in the ownership voucher.

The innovation of the ownership voucher is that it permits the device to be manufactured and distributed without prior knowledge of the server's credentials.

The ownership voucher initially contains a public key owned by the device manufacturer. This public key is also embedded in the device. When the device is sent to a supply-chain partner, the ownership voucher is extended to contain the partner's public key, signed by the manufacturer. This is the digital equivalent of endorsing a bank check to a third party. The partner may further extend the ownership voucher using a digital signature, until eventually it is extended to the server that wishes to onboard the device. This server now uses the ownership voucher, the device credentials, and the FDO protocols to onboard the device.

The signature chain in the ownership voucher forms a trust chain that mirrors the supply chain. The logic is that the same trust that allows the physical device to be used with its base configuration also permits it to be onboarded. Conversely, if the device fails to onboard, it is returned in failure through the same supply chain. The ownership voucher signature chain was first proposed by Ernie Brickell at Intel Corporation.

The ledger of signatures in the ownership voucher allows the device to be routed from supplier to supplier on an "as determined" basis. A distributor can stock many units of a device, then send units to other distributors or end customers as they are ordered by extending the ownership voucher signatures to the shipping target. Extending the ownership voucher requires that public keys be provided or kept on file for expected product destinations.

To help understand this new data structure, a couple of "building rules" for the ledger are helpful. For a given subsegment of the supply chain:

A. Where every entity independently decides the next destination of the device, the ownership voucher's ledger includes a key for each entity.

B. Where an arbitrary set of entities always routes the device to one destination, the ownership voucher's ledger contains only that one destination.

Rule A applies to resellers, who extend the supply chain as buyers appear.

Rule B applies to delivery services and predetermined supply chains.

Rule B also applies to the entire supply chain when all devices onboard to a single cloud service. In this case, a trick is possible, where the cloud service assigns each tenant a key for FDO. The device manufacturer extends each ownership voucher to the key to the purchasing tenant instead of unboxing the device to insert a tenant token. When the device onboards, the key in the ownership voucher identifies the tenant.

A cryptographically random identifier of 128 bits is chosen as a device identifier for FDO, or globally unique identifier (GUID), and is stored in both the device and the ownership voucher. This serves as an identifier for protocols, making it easier for the server to choose the correct ownership voucher for a given device. The randomness makes it hard for an attacker to predict the GUID value. The GUID lasts until the device is successfully onboarded; it is then replaced with a new one.

FDO protocols. FDO specifies four protocols:

▸ **Device Initialize (DI).** Permits storing the FDO device's credentials using a protocol interaction. For some devices, it is more convenient to store FDO credentials directly, such as in flash memory. This is also permitted.

▸ **Transfer Ownership 0 (TO0).** Allows the server to register its IP address and/or DNS name with a rendezvous server, explained in the next section.

▸ **Transfer Ownership 1 (TO1).** Allows the onboarding device to query the IP address and/or DNS name from the rendezvous server.

▸ **Transfer Ownership 2 (TO2).** Allows the onboarding device to authenticate and onboard to the intended server.

In special environments where the intended server can be found directly, such as using network broadcast, the TO0 and TO1 protocols may be skipped, and the TO2 protocol stands alone.

Network connection. As an application onboarding protocol, FDO does not specifically address the network layer connection. For an unauthenticated, wired Ethernet, the installer plugs in a connector. For Wi-Fi networks, FDO uses a Wi-Fi segment with fixed parameters as an onboarding network. This can be restricted from general access. Alternatively, the FDO device can be attached to the Wi-Fi network manually.

Mechanisms to automatically attach FDO hosts to a Wi-Fi network have been proposed, which is an area of active development for FDO.

Device and server rendezvous (TO0 and TO1 protocols). When the FDO device is powered on for the first time, it has its FDO credentials but no information about the network to which it is connected or the identity of the server. Finding the correct server is its first task. The "correct" server means the one with the ownership voucher having the device's GUID. This is accomplished using a rendezvous server and the TO0 and TO1 protocols.

The ownership voucher and device credentials include instructions to find a compatible set of rendezvous servers. The prospective server uses the TO0 protocol to register its GUID against its address (for example, IP address and TCP port) with a rendezvous server. The device uses its GUID in the TO1 protocol to obtain the address of a prospective server. The device then uses the TO2 protocol to authenticate directly to the server.

The rendezvous protocol is similar in concept to a DNS lookup but has some advantages:

▸ The server can autonomically register itself with the rendezvous server using information in the ownership voucher.

▸ Since the rendezvous server is an application server, no special rights to the network are needed.

▸ Rendezvous works the same using an on-premises server as on the Internet.

▸ Rendezvous works in a closed network, if the server, device, and rendezvous server are all accessible to each other.

An escape allows network-specific mechanisms to substitute for the rendezvous protocol. For example, a corporate network could use the DNS Service Discovery (DNS_SD) mechanism to find the FDO owner.[4]

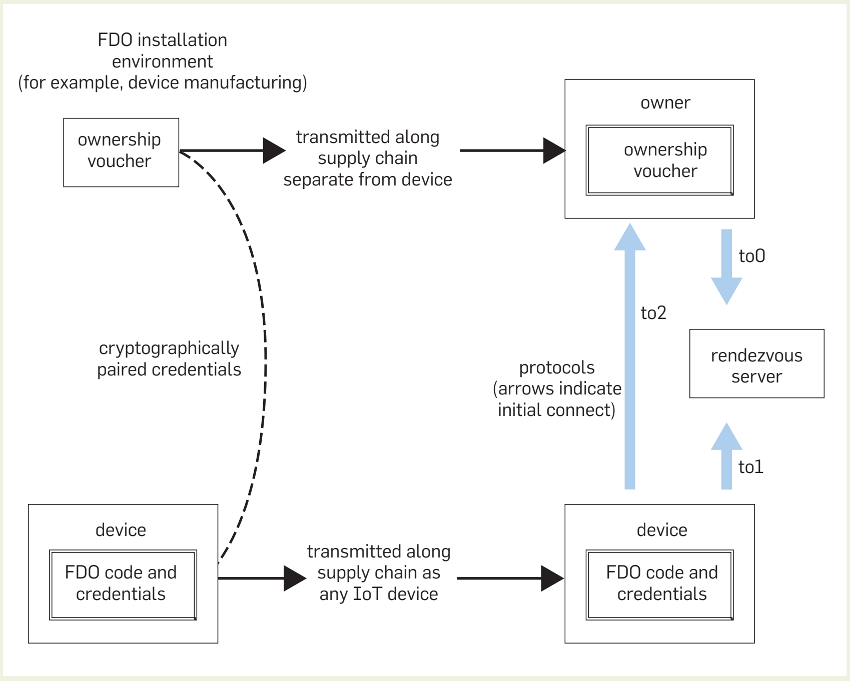
Onboarding using the TO2 protocol. An installed device starts by determining the location of a prospective server using the rendezvous protocol. Then it uses the TO2 protocol to attempt to onboard to this server. If the TO2 protocol is unsuccessful, the device tries other rendezvous servers (if any), then delays and repeats the process.

The TO2 protocol starts with a mutual authentication of device and server. The device is authenticated by generating an attestation based on the device key. The attestation uses the EAT and the configured device key. The server can verify the EAT message using the device's certificate, available in the ownership voucher.

The server authenticates using the ownership voucher by "walking" the ownership voucher signature chain and arriving at the public key of the server itself. (See the description of the ownership voucher noted previously.) The server then signs a challenge to authenticate itself and complete mutual authentication.

Next, a key exchange, usually an ECDH operation, is performed to cre-

Figure 2. Untrusted installer.


FDO installation environment (for example, device manufacturing)
ownership voucher
transmitted along supply chain separate from device
owner
ownership voucher
to0
to2
cryptographically paired credentials
protocols (arrows indicate initial connect)
rendezvous server
to1
device
FDO code and credentials
transmitted along supply chain as any IoT device
device
FDO code and credentials

Comparison Summary

- Both trusted and untrusted installers provide efficient ways to onboard devices.
- Both add overhead to the overall process:
  - The trusted installer adds a small overhead to each install.
  - The untrusted installer adds a small overhead to each player in the supply chain.
- Both rely on specific trust relationships:
  - The trusted installer can go astray if other persons with less trust can take on the installer role unexpectedly.
  - The untrusted installer can go astray if supply-chain entities are not careful with their digital signatures.
- The trusted installer is most efficient for a small number of installs but can be a burden when many devices are installed at once.
- The untrusted installer is equally efficient for any number of installs but requires special handling in the entire supply chain for the ownership voucher.

ate a shared secret across the connection. The shared secret is passed through a key-definition function to obtain credentials to create an integrity-verified, encrypted tunnel. In the actual FDO protocol messages, authentication and key-exchange operations are packed together to allow the authentication message to also authorize the key exchange.

Once the FDO encrypted tunnel is complete, the protocol enters the onboarding phase. Since FDO is intended for general-purpose application onboarding, the behavior of this phase is flexible. The connection transitions to allow subprotocols to run, called FDO Service Info Modules (FSIMs). Since they run only over the encrypted tunnel, the FSIM actions are exclusive to the server and device and permit the server to configure credentials into the device with cryptographic privacy and security.

The base set of FSIMs provide:

- **File download,** including downloading scripts for installation.
- **Command execute** to invoke scripts and make changes to the device.
- **File upload,** to retrieve command output.
- **File download from a URL using HTTP.** This method is faster than file download over the FDO tunnel, but the tunnel can provision credentials for verifying such files.
- **PKI provisioning using a certificate signing request** (CSR).[9]

For devices with limited command-processing capabilities, custom FSIMs can be added to perform firmware updates or access devices such as Secure Elements or Trusted Platform Modules (TPMs).

Conceptually, the server owner develops and maintains a script for each kind of device that invokes FSIMs. As FSIMs become more standardized, it is likely that classes of devices will be onboarded using a single script, perhaps based on operating system (for example, base Linux functions).

Devices with special hardware or firmware need custom commands; these can be abstracted using FSIMs that are attuned to this class of hardware. For example, the CSR FSIM handles the key-provisioning functions of both a TPM and secure element for onboarding. The evolution from specific to generic device characteristics is in the early stages for FDO.

A failure during onboarding causes the device to run FDO again. Recovery from a partial onboard is left up to the server; idempotent operations within FSIMs can improve robustness after partial onboarding.

When all onboarding operations have completed successfully, the server downloads replacements for critical elements of the device credentials. This invalidates previous copies of the ownership voucher. The device responds with information to create a new ownership voucher, which the onboarding server alone possesses. Then the device sets a flag indicating that onboarding has completed, completes the protocol connection, and reboots itself. When the device comes up, the flag causes it to bypass FDO processing, and the device goes into service based on the results of the FDO onboard.

Since FDO ends with a new set of device credentials and matching ownership voucher, the FDO protocol can be used again as needed. For example, FDO can be used to repurpose the device some months later, using the new ownership voucher, but it has no independent function for preparing a device to be repurposed. This function must be handled by the device-management mechanisms before FDO is reenabled.

Comparison of Trusted and Untrusted Installer Techniques

The trusted and untrusted installer techniques exhibit different tradeoffs for installing devices. Here, we discuss the usefulness of each technique. Both trusted and untrusted installer techniques allow complex devices to be installed automatically, and both solve the basic problems of onboarding:

- Ensuring the device is connected to the correct server.
- Establishing trust between device and server.
- Downloading credentials or other data items into the device.

For both trusted and untrusted installer techniques, establishing trust in the device is done using a digital signature with a key in the device. The key, if stored securely, establishes the identity of the device within the context of a public key or certificate from the supply chain. It is the author's experience that a device manufacturer's key and certificate is the most accepted by users, even for a resale situation. Perhaps users assume the quality of the device is based on the reputation of the manufacturer, and that reputation extends to the certificate.

In Wi-Fi Easy Connect, the public key of the device is conveyed to the server by the trusted installer. The protocol keeps onboarding information from the installer private, so the security of the onboarding is unaffected. The trust of the installer affects the accuracy of the device selection. If the environment has individuals who are less trusted but still have access to the configurator, this can affect the security of the result. For example, if friends or visitors have access to a home Wi-Fi network (and the configurator), they could impersonate the installer and bring devices into the network.

As mentioned earlier, server authentication is an option for Wi-Fi Easy Connect.

On the untrusted installer side (FDO), the ownership voucher has an explicit trust mechanism tied to the server that prevents the attacker from impersonating the trust of the server. In this way, FDO extends the trust of the supply chain to identify the server, at the cost of transporting a new object through the supply chain.

For ease of installation, FDO requires no protocol actions on the part of the installer. The trusted installer protocols require specific installer actions, but these are efficient and simple.

In a large installation, it may be difficult to sequence trusted installer actions in parallel. When many devices are onboarded, the time spent by the trusted installer adds up. A trusted installer installing 1,000 devices spends a lot of time scanning QR codes. The Wi-Fi Easy Connect specification has the option to combine multiple QR codes (for example, on a package of "smart" light bulbs). However, this technique has not yet scaled to commercial installation levels.

In contrast, the untrusted installer can onboard devices in parallel, and the devices onboard autonomously as they are powered on. The installer time overlaps the installation time.

Operation in a closed network (no Internet access) is another important issue. Direct Internet access is not actually a requirement of any of the onboarding techniques presented. In Wi-Fi Easy Connect, all communications are with the Wi-Fi network. FDO can onboard when the device, server with ownership voucher, and rendezvous server are mutually accessible. Table 2 includes a summary of the key differences between trusted and untrusted installers.

Conclusion

Automatic onboarding of devices is an important technique to handle the increasing number of "edge" and IoT devices being installed. Onboarding of devices is different from most device-management functions because the device's trust transitions from the factory and supply chain to the target application. To speed the process with automatic onboarding, the trust relationship in the supply chain must be formalized in the device to allow the transition to be automated.

The two general classes of automatic onboarding protocols are those using a trusted installer (person), represented by Wi-Fi Easy Connect, and those using an untrusted installer, such as FDO.

A trusted installer participates in constructing the trust relationship between the device and the server, using a configurator tool. This participation can be reliable, straightforward, and as trustworthy as the trust in the trusted installer. The onboarding guarantees can be subverted, however, if less-trusted people have network or facility access or can masquerade as trusted. Trusted installer devices require some action per install, so many installs incur a linear increase in installer overhead.

In the untrusted installer case, the installer may place or attach the device, but plays no part in transitioning trust. This requires an external mechanism. In FDO, that mechanism is the ownership voucher and must be passed between supply-chain entities with authorizing digital signatures. Supply-chain partners must be chosen carefully, since the FDO trust relationship relies on their accurate processing of ownership vouchers. After the supply chain has made this additional effort, the onboarding itself has no per-device overhead, and devices may onboard in parallel. C

Table 2. Untrusted and trusted installers.

	Trusted installer (Wi-Fi Easy Connect)	Untrusted installer (FDO)
Protocol entity names (roles: server, device, installer, configurator)	Server, device, installer (person), Configurator	Server, device; no installer role in the protocol. Rendezvous server also used
Trust of device	Asymmetric key pair.	Asymmetric key pair, manufacturer's certificate
Trust of server	Installer conveys trust, may also use PKI trust	Ownership voucher conveys trust, may also use PKI trust
Trust of supply chain	Required, to ensure credentials are not compromised	Required, to ensure credentials are not compromised
Changing credentials to re-run	Network configuration	Automatic when you run TO2 protocol
Supply chain overhead	Low, credential shipped in box with product	Ownership voucher must be passed along separately, may need to be extended in supply chain
Home premises	Good	Possible if fulfillment partner keeps track of ownership vouchers
Corporate premises	Small installer overhead per device tends to linear scaling of time for many installs	Good. No installer overhead per device, onboard devices in parallel
Closed network	Yes, all needed credentials must be in the network	Yes, all needed credentials must be in the network

References

1. Barker, E. et al. Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography. *NIST, Computer Security Resource Center* (Apr. 2018); https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final.
2. Bormann, C. and Hoffman, P. Concise Binary Object Representation (CBOR), STD 94, RFC 8949. *RFC Editor*. IETF, 2020; https://www.rfc-editor.org/info/std94.
3. Chen, L. Recommendation for key derivation using pseudorandom functions (revised). *NIST Technical Series Publications*, National Institute of Standards and Technology, 2022; https://bit.ly/3TntdqQ.
4. Cheshire, S. and Krochmal, M. *DNS-based service discovery, RFC 6763*. IETF, 2013; https://www.ietf.org/rfc/rfc6763.txt.
5. Cooper, D. et al. *X.509 Certificate and Certificate Revocation List (CRL) profile, RFC 5280*. IETF, 2008; https://www.ietf.org/rfc/rfc5280.txt.
6. Cooper, G. et al. *FIDO Device Onboard Specification 1.1*. FIDO Alliance, 2022; https://bit.ly/3tqfrZY.
7. FIDO Alliance. *Home FIDO Alliance Specifications Overview*. FIDO Alliance, 2022; https://fidoalliance.org/specifications/.
8. Lundblade, L. et al. The Entity Attestation Token (EAT), draft-ietf-rats-eat-24. *IETF Datatracker*, 2022; https://datatracker.ietf.org/doc/draft-ietf-rats-eat/.
9. Nystrom, M. and Kaliski, B. PKCS #10: Certification Request Syntax Specification Version 1.7, RFC 2986. *RFC Editor*. IETF, 2000; https://www.rfc-editor.org/rfc/rfc2986.
10. Schaad, J. and Cellars, A. CBOR Object Signing and Encryption (COSE), RFC 8152. *IETF Datatracker*. IETF, 2017. https://datatracker.ietf.org/doc/html/rfc8152.
11. Sheffer, Y. et al. Summarizing known attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS), RFC 7457. *IETF Datatracker*, 2015; https://www.rfc-editor.org/rfc/rfc7457.html.
12. Wi-Fi Alliance. *Wi-Fi Easy Connect specification*, 2020; https://www.wi-fi.org/file/wi-fi-easy-connect-specification.

Geoffrey H. Cooper is a principal engineer at Intel Corporation and has worked on automatic onboarding for more than five years. Previously, he worked at a security startup, Secure Computing, and McAfee.