IoT Application Provisioning for Security Using FDO and TPM

December 2022

Editors:
Geoffrey Cooper, Intel
Steve Hanna, Infineon
Mario Larouche, Intel

Abstract

The Internet of Things (IoT) offers the appealing possibility of harnessing millions of devices that measure or modify real world things and allows processes to be optimized. There are many opportunities: thermostats optimized for tomorrow's weather, shipping containers optimized for packing before their contents arrive, assembly lines that reconfigure automatically for each item, street lighting adjusted to make streets safer, and so on.

However, to get the benefit of IoT, we need to install and provision millions of devices so they are identified and controlled by their owners over the internet, or controlled using internet technology on a private network. This has been a painful process and has slowed down the progress of IoT. Now, a group of companies within FIDO Alliance is introducing a new automatic onboarding mechanism with the potential to make the manufacture-to-deployment phase of IoT both fast and secure.

FIDO Device Onboard (FDO) is a set of protocols from FIDO Alliance that uses a technique called "late binding" to allow the security of the manufacturing environment to extend towards securing the automatic provisioning of IoT devices when they are first deployed. Late binding allows a single manufacturing credential in the device to be used to onboard the same device into different IoT ecosystems without requiring separate manufacturing targets (SKUs) for each ecosystem or updating software manually as a device is provisioned. Using FDO, two customers with different IoT managers can buy two of the same IoT devices, and each will onboard securely and automatically to their respective owners' management systems, regardless of where they are deployed.

Contents

Tables

1. Device Onboarding for Purpose-Built Devices

The influence of IoT is limited by our ability to deploy IoT devices into the field with complete confidence in their security, controlled by a remote manager IoT Platform (either cloud-based or local network-based). Manual onboarding techniques have been used for early IoT deployments, but they are hard to scale. An installer who logs into the IoT device to set parameters may inadvertently change other parameters on the device. The installers also might be able to see or modify private keys or other critical data, which can be a trust issue. A small mistake by the installer might install the wrong credentials on a device, confuse two devices in a long list, or pick up the wrong USB stick.

The device console can also be accessed later by attackers who happen to have physical access to the device. This risk scales with the number of devices and the length of time a device is deployed. Untrusted applications might be downloaded or started from the console and might persist as permanent unauthorized services. Critical security settings might be changed, allowing other attacks later.

Staging the devices in a secure location with trusted staff, before the devices are installed, makes it easier to vet and train the staff to perform device onboarding; however, there is still the intrinsic risk of a login that might be abused later. Also, staging devices is still slower and more expensive than manual installation.

Conceptually, we could provision keys and other needed device configuration into the devices during manufacturing. This removes the installer from the error-prone manual process of transporting configuration information and makes installation faster and more reliable. Secret keys are created and protected in the device hardware, making trust easier. Once manufacturing is complete, the configuration interface may be removed from the IoT device, so console-based attacks are less likely or impossible. An IoT device that needs no human interaction in its normal deployment can completely remove the user interface.

In the case of purpose-built devices, deployment of keys and other configuration in manufacturing is easy to understand. The device key is allocated during the latter stages of manufacturing. The IoT Platform's key (and IP address) is programmed into the device, and the device key is sent to the IoT Platform, along with some other identifier for the device (e.g., a device GUID). Then the installer can simply plug the IoT device into a power and network source, and the device will function with the IoT Platform automatically.

Still, factory installation has its problems. If symmetric keys must still be transported to the IoT Platform from the manufacturer, a secure path is needed to transport the keys. This requires a secure path through the supply chain, and diligence with the device logs to determine if the symmetric keys have been compromised.

Asymmetric key pairs provide a better solution for purpose-built devices. This is especially the case if the device creates its key pair internally and only exposes its public key, which can be sent down the supply chain so long as accuracy is assured. Also, the IoT Platform is authenticated using its public key, which does not need to be concealed. Many commonly deployed processors have secure key storage mechanisms built into the hardware, which are appropriate for storing cryptographic keys provisioned either on or off the device.

The Trusted Platform Module (TPM) also provides an excellent mechanism for creating asymmetric key pairs to identify IoT devices. IoT Platform keys may be created in the module itself. The TPM hardware can be hardened against multiple attacks--physical, side-channel, or cyber—at a lower cost than the main CPU.

TPMs can also be provisioned with keys and certificates during the TPM manufacture. In this way, IoT Platform keys can be made available to an IoT device when the TPM part is added to the board, with no change to board level manufacture. By bundling keys and their cryptography on the TPM with the secure key stored, the TPM is never required to expose private keys to the main CPU. This makes it impossible for a software attack to expose or copy a permanent credential from the TPM. A successful software exploit can still *use* the keys on the TPM while it is active but cannot steal these keys and bring them to other devices.

2. Automating Device Onboarding for General Purpose Devices

Purpose-built IoT devices have their place, but most IoT devices are more general purpose, where the IoT Platform may be one of various software platforms with various addresses and keys. The keys for the IoT Platform (and its IP address) are not known during manufacturing. For very large IoT Platforms (e.g., cloud based), we could establish device SKUs that differ only in IoT Platform keys and IP addresses, but this is also expensive.

We would like to use the good ideas in TPMs to provision keys for these general-purpose devices, preferably without needing to manually install or stage the devices. The IoT device installation would proceed as follows:

1. Customer buys an IoT device and gets a receipt sent to their IoT Platform identifying the device (perhaps out of the distributor's stock). The receipt refers to keys in the device, such as stored in the device TPM.
2. Installer (person) mounts the IoT device, connects it to a power source and to a network, and powers it on.
3. IoT device determines the correct IoT Platform (the "cloud" or device manager that will manage the device remotely) and authenticates to it based on credentials sent with the receipt step [1]. The device and IoT Platform negotiate all credentials and data needed by the device so that the device starts to provide useful data as part of the IoT Platform's application.

This is an *automatic onboarding* of the IoT device application. Notice that the device is manufactured without a particular IoT Platform target (in the first step). This means any credentials or data (or software) needed to customize the device to a given IoT Platform must be configured automatically, on demand, during the last step of onboarding. This is called "late binding."

3. FIDO Device Onboard: Automatic Onboarding with Late Binding

FIDO Device Onboard (FDO) provides automatic onboarding with late binding, making it possible for a single general purpose IoT device (i.e., one "SKU") to onboard on the IoT Platform of the customer's choice. FDO is designed to provide the IoT device owner and installer with the 1-2-3 step automatic experience described in the previous section.[1]

FDO works by creating credentials for the device during manufacturing. Some of these credentials are stored in the device, and some are sent along with the device through the supply chain. The supply chain credentials are bundled into a textual data structure called an "Ownership Voucher." As the device proceeds through the supply chain, powered off in its shipping box, the Ownership Voucher is extended to create a ledger of ownership. This ledger of ownership permits supply chain entities to warehouse and ship devices where they are needed while maintaining information for automatic onboarding. Later when the IoT device is ready to be installed, the Ownership Voucher is used to authenticate the IoT device to the IoT Platform, and the IoT Platform to the IoT device.

When a new IoT device is installed, installation begins when the IoT device connects to the IoT Platform. It is critical that IoT device connects to the correct IoT Platform. To do this, it must discover the internet address (IP address) of a prospective IoT Platform, connect to it, and authenticate.

[1]The specification for FIDO Device Onboard is available here: https://fidoalliance.org/specifications/download-iot-specifications/.

In FDO, the device determines the prospective IoT Platform using a **Rendezvous Service**, identified in the Ownership Voucher and in the device. The IoT Platform sends the Ownership Voucher to the Rendezvous Service to identify itself as a prospective owner for the device. The Rendezvous Service stores the IoT Platform address information for an interval, until the IoT device contacts the same Rendezvous Service and asks for a prospective owner. Then the Rendezvous Service shares the IoT Platform IP address with the device. The Rendezvous Service never sees the device/IoT Platform negotiations; it just gives advice about the address. The Ownership Voucher can allow the device to contact multiple Rendezvous Services, including a service within the same closed network as the device.

Next, the device contacts the prospective owner directly, the device using the stored device credentials, and the prospective owner using credentials in the Ownership Voucher. The device credentials and Ownership Voucher are cryptographically mated, allowing the device and prospective owner to mutually authenticate and create an encrypted tunnel between them.

Consistent with **late binding**, the encrypted tunnel permits the now-authenticated owner to create new credentials and data in the device. This allows the device to accommodate new credentials known only to the device and its newly established owner. The owner can also cause the device to invoke local or network commands, choosing the right network services and applications to run on the device. If the device has a software update capability, new drivers or applications can be added to the device.

In this way, the device starts onboarding using the credentials provided in the factory and finishes with new credentials, ready to operate in its intended application. This all happens in the first few minutes of the device's installation.

Late binding is a flexible mechanism allowing symmetric and asymmetric keys, passwords, data items to be installed, and specific actions and even some software updates to run. Late binding makes use of the underlying operating environment. For example, a TPM can be used to generate application keys and establish trust during late binding.

4. FDO Late Binding Mechanisms (ServiceInfo)

FDO is equipped with a special mechanism called “ServiceInfo” to enable a given IoT Platform to interact with a target IoT device.

During the late binding step, described in the previous section, software from the IoT Platform can interact with software from the IoT device over the FDO encrypted channel. Software on the IoT device exports late binding modules to the local FDO implementation, which are offered to the IoT Platform. The IoT Platform interacts with modules using messages.

FDO implements only a single module, “devmod,” which is used by the device to list available modules to the IoT Platform. We expect individual devices or program operating environments will export and document their own modules to handle their special features over time.

To illustrate how ServiceInfo late binding works, a message interaction to store a new root password might look like this:

#	From/To	Message	Comment
M1	Platform→Device	Password-Manager:user operator	Select user
M2	Platform→Device	Password-Manager:pass 41076e9f33384ae	Set password
M3	Platform→Device	Password-Manager:user root	Select user
M4	Platform→Device	Password-Manager:login-disable	Prevent root login

Table 1– Late Binding with ServiceInfo

Here **Password-Manager** is a module in the device, which is invoked by the IoT Platform. First, the password for "operator" is set (M1, M2), then the "root" account is disabled (M3, M4) to prevent mistaken or malicious logins to this account. All these commands are protected by the FDO encrypted tunnel, so the password (M2) is seen only by the IoT Platform and the device itself.

Late binding can also be applied to cryptographic keys. For example, a TPM on the IoT device can create an asymmetric key pair (an associated public and private key), then the IoT Platform can assign trust to it using a CA certificate:

#	From/To	Message	Comment
M5	Device→Platform	AsymKey:csr 1 secp256r1 *PKCS#10-csr*	Send certificate signing request, including public key
M6	Platform→Device	AsymKey:cert 1 *x509cert-in-base64*	Obtain X.509 certificate for the public key, store to slot 1

Table 2 – Late Binding with Cryptographic Keys

In the first message (M5), the device **AsymKey** module has created an asymmetric key pair using elliptic curve cryptography (ECDSA NIST P-256) and sends a Certificate Signing Request (CSR) in PKCS#10 format. The IoT Platform component requests a certificate based on the CSR from a trusted certification authority, then sends it back to the device (M6). The device stores the certificate for later use, such as in future TLS connections to the IoT Platform.

As a more general mechanism, FDO late binding can run scripts to install software. For a Linux-based IoT device, using shell commands is a powerful mechanism. Often a shell command or shell script is created to allow deployed machines to be installed by hand. FDO can be used to automate this step by invoking the same script. The following illustrates this function:

#	From/To	Message	Comment
M7	Platform→Device	fdo_sys:filedesc *filename*	Create file
M8	Platform→Device	fdo_sys:write *filedata(base64)*	Append data to the file
M9	Platform→Device	fdo_sys:exec argv(base64)	Use the shell to execute a file

Table 3 – Late Binding with Scripts

In the **fdo_sys** module, the first message (M7), creates a file local to the device. Subsequently, one or more fdo_sys:write commands sends textual or binary data to the file (M8). As before, the file data is downloaded via the FDO encrypted channel, so it is confidential between the IoT Platform and the device being onboarded. The fdo_sys:exec message (M9) executes the file with a given set of command line arguments.

The messages M7, M8, and M9 indicate how to download a file and execute it using the shell. The commands can be executed multiple times to download multiple files.

Using a shell script to install a device is a common practice in IoT deployments, and this example illustrates how an existing script can be adapted to automated use with FDO. FDO is not itself a software-update mechanism, but it can invoke software update mechanisms within the device operating system to achieve this effect. For example, a shell script file can use networking commands such as Linux **wget** to download software package upgrades. Using this technique, it is possible to upgrade a Linux system to choose client software for a given IoT Platform, or even download a new set of client software not previously on the device. Most Linux distributions have a system command that upgrades one or more modules automatically; another FDO module can also serve to abstract the differences between these upgrade commands.

Late binding automates device installation and allows a device to integrate to a new IoT Platform, a new authentication mechanism, or to new configuration data needed to calibrate sensors. Work is still required for an IoT Platform to accommodate a new device, similar to the work needed to manually install the first instance of this same device. However, once this is accomplished, FDO allows many devices to be onboarded with great speed, security, and reliability.

By installing modules, FDO can both use and allow the IoT Platform to use special cryptographic hardware to increase the flexibility and security of the IoT solution. In the next section, we learn more about a common mechanism for integrating cryptographic key security into a hardware platform, the Trusted Platform Module or TPM.

5. TPM and How It Adds Security and Convenience

The Trusted Platform Module (TPM) is a widely adopted cybersecurity chip used around the world. The popularity of TPMs is explained by their ability to provide many strong security features implemented in a separate execution environment with a standard interface at an affordable price. For this reason, most PCs and servers include a TPM, and an increasing number of embedded systems do also.

For IoT devices, FDO and TPM are a perfect match. FDO provides easy and secure installation of the device into the customer's environment. The TPM secures the FDO credentials, reducing the risk that they could be compromised, while simultaneously simplifying the manufacturing process.

During FDO onboarding, ServiceInfo modules can be used to create new trusted keypairs in the TPM, to be used as application keys during device operation. This separates the cryptography for device operation from any keys that might have been stored during manufacturing or during the supply-chain. If a device is repurposed, the new application keys separate the new function from the old environment.

In addition to downloaded keys, FDO can be used with the TPM to establish certificate trust for a key that exists *only* within the TPM. This provides the strongest level of device security since the private key can never be used outside the TPM. In other words, if a person still owns the device, they still own the key.

When a new application key pair is requested using ServiceInfo, the TPM creates a Certificate Signing Request (CSR) data structure and transmits it to the IoT Platform, again using ServiceInfo. The IoT Platform transmits the CSR to a known Certification Authority to generate a certificate that is trusted by the IoT Platform, then it sends the certificate to the device, again using ServiceInfo. The device can then use the certificate with the private key in the TPM to perform Transport Layer Security (TLS) connections to the IoT Platform. The certificate is known to the [1] the device, [2] the IoT Platform, and [3] the trusted Certification Authority that created it. The key is known only to the device.

We can see from this example that the TPM and FDO cooperate to help each other be secure. To fully understand how FDO and TPM work together, it helps to understand some of the key features of the TPM:

Cryptographically secure random number and key generation

Modern cybersecurity measures such as FDO require the ability to generate cryptographically secure random numbers and keys. All certified TPMs include this feature based on a strong source of entropy, thus avoiding the weak, random numbers common in embedded systems.

Secured storage

Secrets such as keys must be stored in a secured location to prevent attackers from gaining access to the secrets. Certified TPMs protect stored secrets against a wide variety of attacks, extending even to protect against compromised application or operating system software, as well as physical attacks and side channel attacks.

Highly sensitive secrets such as private keys can be generated on the TPM and marked with a policy that prevents them from ever leaving the TPM.

Cryptographic operations

Cryptographic operations, such as signing and verification, are implemented in TPMs so that cryptographic keys need never leave the TPM. Certified TPMs are tested to verify that performing these cryptographic operations does not unintentionally reveal the keys through any channel, including radio frequency emissions or power usage.

Trusted computing

The TPM's special trusted computing features permit a cloud service to remotely verify the version and integrity of software running on an embedded system that includes a TPM. They can also be used in TPM policies to restrict access to protected data stored on the TPM.

An implementation of FDO can directly use the first three of these TPM features to provide strong protection for the device keys. By generating the keys in the TPM and setting a policy that prevents releasing the private key from the TPM, strong protection against device cloning or impersonation is provided. TPM's secure storage of these secrets also protects against tampering in the supply chain. Even the complete compromise of the device's operating system or application code cannot result in disclosure of the protected keys and data in the TPM.

To support trusted computing, the TPM produces an attestation of system integrity. This attestation may be generated and retrieved securely during FDO by creating a ServiceInfo module for the TPM. A secured attestation allows the IoT Platform to determine the correctness of the device firmware and OS *before* it decides to complete onboarding of the device. If the results are not as expected, the device may still be onboarded, but directed into a dormant state until system administrators can fix it.

FDO parameters may be stored into the TPM:

The device private key and HMAC secret need confidentiality protections. These credentials can be stored on the TPM to protect them against disclosure to unauthorized parties.

Other FDO configured values such as the manufacturer's public key are not secret but do need protection against unauthorized modification. By storing these values in a TPM with a policy that makes them read-only, they can be protected against unauthorized changes either through tampering or through software compromise.

When FDO parameters are stored within the TPM to a protected local non-volatile memory (NVM), the software on the device can be upgraded without worrying that the data stored on the TPM will be lost. Even if the device's NVM must be wiped or replaced, the TPM will retain its protected data. This feature can be used to enhance device resiliency. A "rescue" operating environment resident only in RAM can restore all device storage, based on FDO parameters stored in the TPM.

It is even possible to store FDO parameters in the TPM *before* a device is manufactured. The TPM manufacturer can embed FDO parameters, so that a device-level manufacturer can manufacture the device pre-configured for FDO.

6. Conclusion

Device onboarding is a major concern for the efficient deployment of IoT devices. Automatic onboarding through FDO can accelerate deployments, especially when many devices are deployed at once. Using late binding, FDO devices can target multiple target sites with a single device SKU, allowing for a more efficient manufacturing supply chain.

TPM, a hardware module that provides secure credential storage and trusted implementations of cryptography, combines well with FDO. TPM can store credentials for FDO securely, and it provides the basic asymmetric cryptography for FDO's authentication. FDO, in turn, can work with TPM to provision application keys and certificates into TPM, allowing customers to have private key materials with chains of trust matching their organizational requirements. TPM can potentially also be used to store FDO factory credentials, allowing devices to adopt FDO without a change to the manufacturing line.

7. Appendix

Credential	Size	Security	Description
Protocol version	1 byte	A,I	Version of FDO
Crypto suite info	1-2 bytes	A,I	Key type, encoding, hash type, hash size
GUID	128 bits	A,I	Unique ID used to identify device before onboarding
RendezvousInfo	Estimate 40-80 char	A,I	String containing information on how to find the rendezvous service; typically a few verbs and DNS names.
Hash of Manufacturer's Public Key	256 or 384 bits	A,I	Hash of the base key used in the Ownership Voucher. This binds the device to the Ownership Voucher. The manufacturer does not need to maintain or use this key after the Ownership Voucher is transmitted.
Device Identification String	Estimate 40 char	A,I	A string containing the device model name/number, a hint to marshal device parameters in the IoT Platform (Device Manager)
HMAC secret	256 or 384 bits	C,A,I	A secret to the HMAC in the Ownership Voucher. This is used to bind the base of the Ownership Voucher to the software instance of the device. It is intended that this secret is known only in the device (or a TEE within the device)
Device Key	ECDSA 256/384	C,A,I	Device private identification key. The device certificate is in the Ownership Voucher and need not be locally stored to use FDO

Table 4 – FDO device – Resident Credentials

8. References

"Download IoT Specifications." FIDO Alliance. Accessed May 12, 2022. https://fidoalliance.org/%20specifications/download-iot-specifications/.

"FIDO Device Onboard Specification 1.1." FIDO Alliance. April 19, 2022. https://fidoalliance.org/specs/FDO/FIDO-Device-Onboard-PS-v1.1-20220419/FIDO-Device-Onboard-PS-v1.1-20220419.html#informative.

Kaliski, Burt, and Magnus Nystrom. "Certification Request Syntax Specification, v. 1.7." The Internet Society, November 2000. Republication of PKCS #10 v1.7 from RSA Laboratories' Public-Key Cryptography Standards series. https://datatracker.ietf.org/doc/html/rfc2986

POC code for FDO is available from the LF-Edge project at: https://github.com/secure-device-onboard

U.S. Department of Commerce. National Institute of Standards and Technology. "Digital Signature Standard." July 2013. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.