C. Jennings
Cisco
February 14, 2012

Transitive Trust Enrollment for Constrained Devices

Abstract

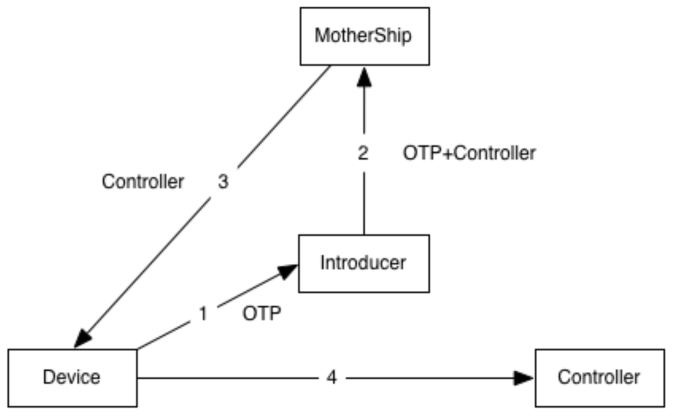
This document provides a very early sketch of a enrollment protocol that allows constrained internet devices to securely enroll into a system. As the work is in its early phase, many details remain to be resolved. The solution is based on the idea that each device will be manufactured with a one time password that can be used by the customer to tell the device which controller to enroll with.

1. Introduction

Secure enrollment of devices into internet-based systems has never been easy. The constrained devices that need to be enrolled into systems today face many challenges. Typically, simple devices have no user interface such as a keyboard or screen - they may have only a single button or LED. At the time they are installed, there may not be a working network or even power. However, these devices are being used for applications that are increasingly important and safety-critical, so they need to have reasonable security and privacy characteristics. This documents specifies an enrollment system for such devices.

In many systems, there is a need to configured a Device, such as a sensor or actuator, so that it is controlled by some specific controller. In the case Devices like a switch and light, it may be that all the Controller does is later configure the switch to control the light. To make this happen, both Devices need to be under the control of a common Controller that is authorized to make changes to the Devices.

The simplified high-level information flow is illustrated in the following figure. The goal is to get to the point where the Device knows that it should be talking to the Controller.



When the Manufacturer builds the Device, it includes a One Time Password (OTP) that the Introducer can use to enroll the Device with the Controller. The Manufacturer also runs a website known as the MotherShip that knows the OTP for every device that Manufacturer builds. The Device can include the OTP as a QR code on the outside of the Device. When the Device is installed, the installer uses a software agent known as the Introducer. The Introducer would typically be something like an application running on an iPhone. When the Device is installed, the Introducer can scan the QR code on the Device to find the OTP (Message 1). The Introducer then contacts the MotherShip and uses the OTP to tell the MotherShip which Controller this Device is should use (Message 3). Later, the first time the

Device boots up and gets network connectivity, it contacts the MotherShip, and the MotherShip tells the Device which Controller to talk to (Message 3). From that point on, any time the Device boots, the Device can communicate directly with the Controller (Message 4). The actual message flow is slightly more complicated and shown in **Section 2**, but it uses the same basic idea as this simplified flow.

The system is designed to achieve several desirable properties:

- Can work for Devices with very limited memory and processing power
- Does not require network or power to be up when the Device is installed
- Is fairly secure (see more in the security section)
- Minimal addition to manufacturing costs
- The installer can detect if the OTP has already been used
- Provides a work flow in which a Device does not need to be taken out of the box to be enrolled. This can be very important to enable consumers themselves to enroll devices they buy from a service provider.
- Works with common Firewall and NAT network topologies

One of the key steps in making this system work is getting the OTP from the Device to Introducer. There are several ways that could happen but a few of the approaches considered here are:

- Using a QR code or other bar code printed on the Device and/or box it comes in
- Having a single LED on the Device that blinks out the OTP information and using a video capture application on the Introducer to read this
- The manufacture providing the OTP in some other machine readable form
- Including the OTP in an RFID tag on the Device that can be read by the Introducer
- Having an electrical interface (such as one wire memory) on the Device that can be read by the Introducer

The semantic level information in each message is discussed in **Section 2** and the syntax of the messages is discussed in **Section 3**. The security properties of the system are described in **Section 4**.

2. Enrollment Information Flow

The Manufacturer, Device, MotherShip, Introducer, and Controller are abbreviated M,D,MS,I,C respectively. The Device, MotherShip, and Controller all use CoAP to communicate with each other and thus each have an asymmetric key pair that is used to form the DTLS connections between them. The MotherShip acts as an HTTP server to communicate with the Introducer and Controller. The MotherShip needs a normal certificate to use HTTPS.

It is assumed that the Device may have a NAT between it and the Controller and that the Device is on the inside of the NAT. The MotherShip is assumed to be a generally accessible server on the internet but the Controller and Device can be on the inside of a Firewall or NAT between them and the MotherShip.

In the following message flow we use the following definitions:

Fingerprint
: This refers to a hash of the DTLS public key used by the associated network element. "MS Fingerprint" means a fingerprint of the public key that the MotherShip will use when forming CoAP connections over DTLS.

MS ID
: A 32-bit integer that uniquely identifies the MotherShip. **Section 3.4** explains how to use the MS ID to create a URL that can be used to contact the MotherShip.

Dev ID
: A 32-bit integer that identifies the Device and when combined with the MotherShip is unique. Two Devices that use the same MotherShip cannot have the same Dev ID.

Dev URN
: A globally unique URN assigned by the Manufacturer to uniquely identify this Device. This SHOULD be one of the URNs from **[I-D.arkko-core-dev-urn]**.

OTP
: The One Time Password created by the Manufacturer for enrolling the Device. This is a cryptographically random 64-bit integer.

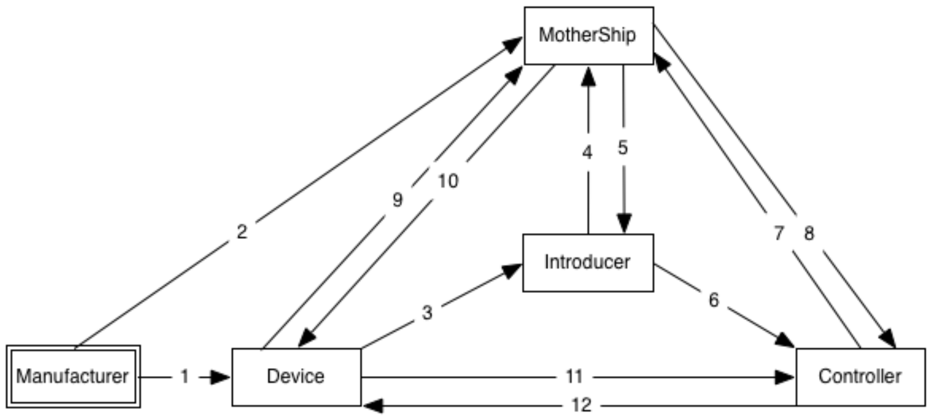
C Addr

Address of the Controller. This is an IPv4 or IPv6 address and port which the Device can use to form a CoAP connection to the Controller.

Dev Descp
: A locally significant string that the Introducer can assign to a Device. For example, the convention for a thermostat in building 30, floor2, office 361 might be assign the string "BLD30/2/361 - Thermostat". This string is provided purely as a way to let the Introducer and Controller exchange information that may be useful for the Installer.

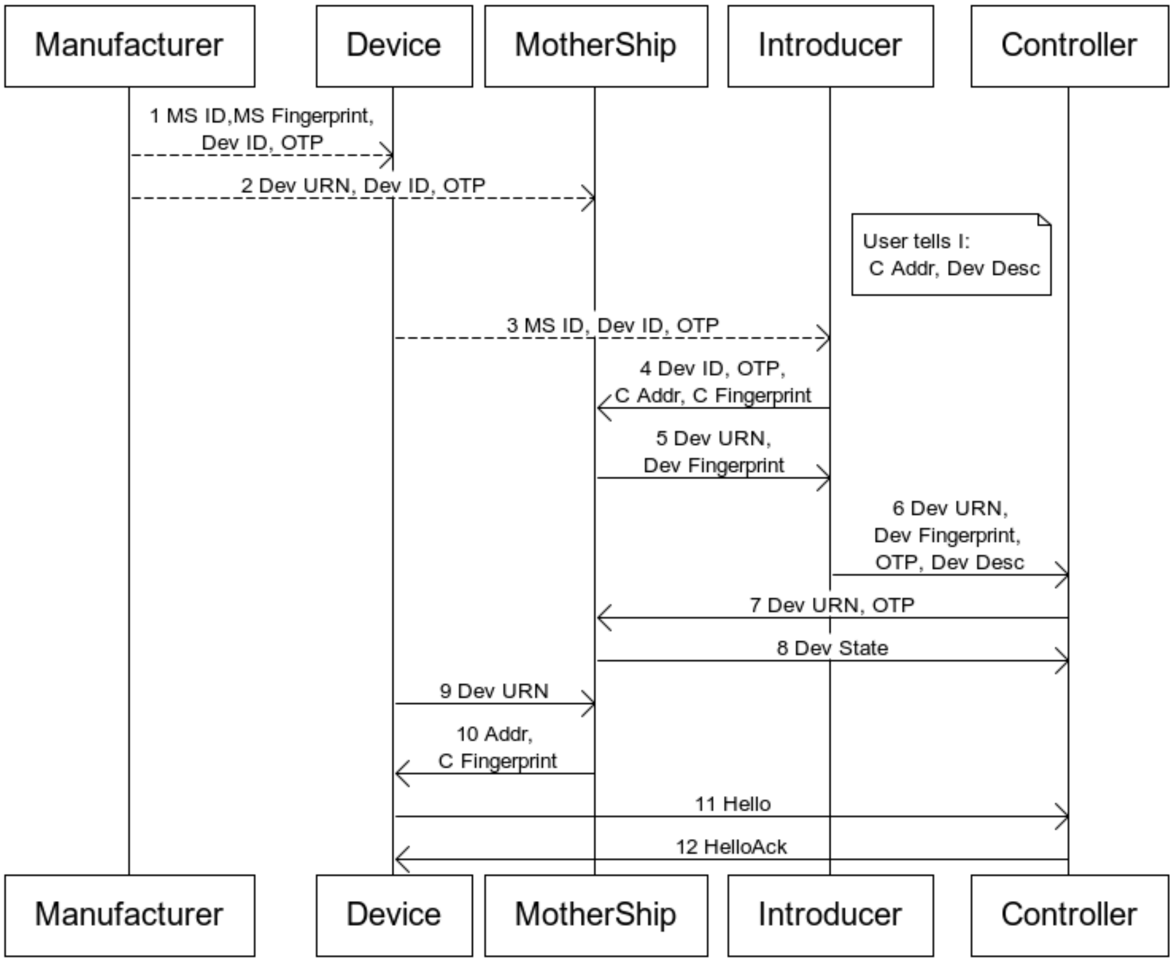
Dev Status
: The Controller can query the MotherShip for the enrollment status of a Device that is enrolled with that Controller. The various states returned are defined in **Section 3.2**.

The information flow is illustrated in the following figure. The goal is get to the point where the Device knows that it should be talking to the Controller, the Controller knows it should be talking the Device, and the Device and Controller can communicate using CoAP and authenticate each other using their public keys.



When the Manufacturer builds the Device, it includes a One Time Password (OTP) on the Device and MotherShip (Message 1 and 2). When the Device is installed, the Introducer reads OTP and other information from the Device (Message 3). The Introducer then uses the OTP to tell the MotherShip which Controller this Device should use (Message 4 and 5). Later the Device contacts the MotherShip and tells the Device which Controller to talk to, information that the Device saves in non-volatile memory (Message 9 and 10). From that point on, any time the Device boots, it can directly communicate with the Controller (Message 11 and 12).

The Introducer has the option of informing Controller about any Devices that it has enrolled with this Controller (Message 6). The Controller can optionally contact the MotherShip to find out about the status of any Devices that it has not heard from (Messages 7 and 8).



When the Device is built, it needs to be assigned a globally unique URN, a Dev ID, and a MotherShip. A single manufacturer MAY operate many MotherShips as each one can only support 16 million Devices. A perfectly reasonable way to generate the Dev ID is to use the least significant 32 bits of the Device URN. The Device needs to be programmed with the IP address and port of the MotherShip along with the fingerprint of the public key that the MotherShip will use in the DTLS CoAP exchange.

The creation of the MotherShip domain name is discussed in **Section 3.4**. The QR code for the Device MUST be an HTTPS URL that points at the appropriate MotherShip and MUST include a URL parameter called "otp" that is set to OTP represented in hexadecimal and MUST include a URL parameter called "devid" that is set to the Device ID represented in hexadecimal. It MUST use the default HTTP port and MUST have an absolute path of /.well-known/tte. As an example, if the MotherShip's domain name was ""tte-000000.net", the OTP was 0x123456789abcdef0 and the Device ID was 0xABCDEF01, a valid URL would be:

```
https://tte-000000.net/.well-known/tte?
        otp=123456789abcdef0,DevID=abcdef01
```

The QR code SHOULD use an error coding level of "H". This would generate the following QR code:

The Introducer reads the QR code found and the Device, then uses this URL to contact the MotherShip in messages 4 and 5. This URL is referred to as the Enrollment URL .

Messages 4 and 5 MUST be sent over TLS, and the Introducer MUST verify that the HTTPS certificate of the MotherShip matches the URL. The Introducer can perform either an HTTPS GET or POST. If the Introducer does a GET, it MUST make an HTTPS GET request to the Enrollment URL and MUST act as a web browser to process returned HTML pages. In the case of a GET, the MotherShip MUST return a web page that allows the user to enter the IP address and port of the Controller as well as the fingerprint of the Controller's public key used in CoAP. If the Controller does not wish to act as a web browser, instead of using the GET, it will use a PUT. When using a PUT, the Controller MUST make an HTTPS POST request to a URL formed by appending three parameters to the Enrollment URL. The parameters are cip, which MUST have the IP address of the Controller; cport, which MUST have the port of the Controller; and cfingerprint, which MUST have the fingerprint of the Controller's Public Key, represented in hexadecimal. If, and only if, the MotherShip successfully stores the address information, the POST MUST return an HTTP 200 response with a JSON string containing the URN and Fingerprint for that Device. The format of this object is described in **Section 3.2**.

Once the MotherShip has successfully stored the Controller's address for a given OTP, it MUST NOT allow that OTP to be used again to store an address for that Device. The OTP can be used after this to query the status of the enrollment as described in **Section 3.2**.

Message 6 is optional and MAY be omitted. As some point after the Introducer has successfully mapped the Device to the Controller, it can send an HTTP or HTTPS request to the Controller to notify it that it can expect to hear from a particular Device. The message formats for this are defined in **Section 3.3**. This does not need happen immediately and the information can be saved so it can be done far in the future. This might happen if Devices were being installed before the Controller was even operational. In other cases it might be done immediately. (TODO - look at in web browser case having MotherShip redirect Introducer to Controller after successful Introduction.) This is done with an HTTP POST to TBD URL with parameters to convey the Device URN and Fingerprint learned from the MotherShip, the OTP password, and a locally significant description string that can be used to help label the Device for management reasons.

In the case where the Controller has learned the URN and OTP for a given Device, it MAY query the MotherShip to find out the enrollment status. It does this with an HTTP GET request to TBD URL. The various statuses that can be returned in TBD JSON doc are: revoked, not mapped, mapped, registered. TODO - could use better names and descriptions.

When the Device has powered up and has network connectivity for the first time, it attempts to form a CoAP connection to the MotherShip. The Device makes a CoAP GET request to TBD URL, passing its URN as a parameter. Details of this message are provided in **Section 3.1**. The Device MUST check that the Public Key provided by the MotherShip in the DTLS connection matches the fingerprint provided by the Manufacturer. The MotherShip needs to look at the Public Key provided in the DTLS and ensure that it matches the fingerprint for this Device that was provided by the Manufacturer. If everything does match, the MotherShip MUST return (in Message 10) the IP address and port for the Controller as well as the Fingerprint for the Controller's public key. Details for the syntax of these messages are provided in **Section 3.1**. If this is successful, the Device MUST store the address and fingerprint for the Controller in non-volatile memory and, on future reboots, skip all the steps before this and connect directly to the Controller. (TODO - Define how retries work if the Device has not yet been enrolled.)

At this point, the Device can form a CoAP connection to the Controller. The Device can verify that it is speaking to the correct Controller by checking that the DTLS Public Key matches the fingerprint for the Controller that was retrieved from the MotherShip. If the Introducer has contacted the Controller in message 6, then the Controller will already have the fingerprint of the Device and can verify that it matches the DTLS information in the connection between the Device and the Controller.

The Controller MAY be configured such that if it does not have the information from Message 6 it can ignore the Device until it gets the information from the Introducer, or, alternatively, such that it can accept the connection based purely on the fact that the network was configured to send messages to the Controller.

3. Message Formats

This section is missing from the current draft and will be completed in future revisions once feedback on the overall design and been incorporated.

3.1. Device Enrollment Query

TODO - define well known COAP URL on MotherShip that the Device uses to get information about Controller.

3.2. JSON Enrollment States

TODO - Define a JSON object with Device URN, Device public key or fingerprint, and enrollment state.

3.3. Controller Enrollment Messages

TODO - define HTTP messages to allow Introducer to tell Controller about a new Device. Need a way for Introducer to tell Controller, the Device public key or fingerprint, the Device URN, and the locally significant label string, and the OTP.

3.4. MotherShip ID and URLs

This system requires a programmatic way to go from a MotherShip ID, which is a 32-bit integer, to an address that can be used to contact that MotherShip. The approach here is to use DNS for that mapping. For a MotherShip ID that has a high order byte of 0x00, the DNS host name of the MotherShip if formed by prepending "tte-" to the lower order 24 bits of the MotherShip ID represented in hexadecimal, and then appending ".net". So the host name for the MotherShip ID 10 would be "tte-00000A.net". MotherShip IDs that have a high order byte other than 0x00 are reserved for future specifications.

A Manufacturer gets a MotherShip ID simply be registering the corresponding DNS entry. The MotherShip ID zero is reserved for examples and MUST NOT be treated as a valid ID by operational systems. A manufacturer wishing to have more than 2^32 Devices would simply register multiple MotherShip IDs.

4. Security Considerations

This section has not really been started and needs lots of work.

TODO - Discuss how one can replace a dead Controller with a new one in an operational network. The short answer is likely that one needs to back up the private keys of the old Controller and move these to the new Controller.

What happens if the OTP is stolen during Device transit? The short answer is that the Device is compromised at this point and needs to be discarded or returned to the manufacture to get a new

Device ID and OTP. The Introducer needs to detect that this has happened and warn the user.

There are additional concerns about Devices that may be operational without ever being introduced to a Controller. For example, if a light switch supported this protocol, but could also be used just as a stand alone light switch, there is a risk the OTP could be stolen by an attacker, with the attacker enrolling the Device to the attacker's Controller. When the correct user installs the light switch, if they never bother to try to Introduce it to anything, they will not detect that it has been compromised. One way to mitigate this risk in situations where it exists might be to include some manual configuration on the Device to indicate that it is to be used in stand-alone mode, such as a jumper that can be cut.

Network topology consideration - Introducer can install firewall rules that allow Devices to contact MotherShip.

why works with NATs / FWs.

5. Variations

5.1. LED Based Enrollment

An alternative to QR codes is to have an LED on the Device flash out the relevant information to the Introducer. The output string is formed by concatenating a 16-bit start of message constant value of 0x0001, followed by the MotherShip ID, Device ID, OTP, and then an 8-bit two's compliment checksum value computed over the previous bytes, including the start of message constant. All values are in network byte order. The resulting string is output using Non-Return-to-Zero Inverted (NRZI) encoding on the LED at a baud rate of 15 bps. This allows a Device such as a smartphone with video capture to detect the signal and recover the information.

TODO - see if this works at 30 bps. See about encoding multiple intensity levels or colors in the LED. Initial experiments indicate this does not work very well as auto contrast in the video camera tends to saturate LED range. Would an Adler-32 checksum be better?

5.2. Bulk Enrollment

Imagine one wants to enroll a whole box of sensors. We should define some scheme where one can simply bar code something on the outside of a box and can bulk enroll all the sensors in the box. Perhaps have a scheme where there is a master secret and start and end Device ID on the outside of box bar code. Then the OTP for a given Device is generated using the master secret and DeviceID of that Device. Need to sort out details of a scheme like this.

5.3. No Public Key Crypto

The examples here assumed that COAP was being used with DTLS with asymmetric keys. It would also be possible to use DTLS in Pre Shared Key (PSK) mode in a very similar flow, where the Introducer provided the MotherShip with the PSK to be used between the Device and the Controller.

6. Open Issues

The references section is in serious need of work - let me know stuff that should be added to it.

Does QR encoding of L work out better than H?

Is there any advantage in having the HTTP URL in well-known space?

Is there some clever way (perhaps zeroconf) for the Introducer to discover the Controller's information?

7. IANA Considerations

TODO - create registry for the top byte of MotherShip ID

TODO register .well-known HTTP URL

8. Acknowledgments

Some of the fundamental ideas in this draft where inspired by Max Pritikin's work. I'd like to thank the following people for review comments: Eric Rescorla

9. References

9.1. Normative References

[I-D.ietf-core-coap]	Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "**Constrained Application Protocol (CoAP)**," draft-ietf-core-coap-08 (work in progress), October 2011 (**TXT**).
[RFC2119]	**Bradner, S.**, "**Key words for use in RFCs to Indicate Requirement Levels**," BCP 14, RFC 2119, March 1997 (**TXT**, **HTML**, **XML**).

9.2. Informative References

[I-D.arkko-core-dev-urn]	Arkko, J., Jennings, C., and Z. Shelby, "**Uniform Resource Names for Device Identifiers**," draft-arkko-core-dev-urn-01 (work in progress), October 2011 (**TXT**).

Author's Address

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA
Phone: +1 408 421-9990
Email: fluffy@cisco.com