Description of a device Bootstrapping and Registration procedure

Jaime Jimenez

March 2016

1 Introduction

The bootstrapping here described is based on OMA's Lightweight Machine to Machine (LWM2M) Bootstrapping [4] and Constrained Application Protocol (CoAP) security modes. In particular we focus on two possible modes, the "PresharedKey" (PSK) and "RAW Public Key" DTLS bindings, leaving "NoSec" and "Certificate" modes aside.

Using symmetric keys has two main advantages. First, by using PSK we avoid the need for heavier cypher-suites such as public key encryption. DTLS is already demanding for resource-constrained environments. Secondly, the shared secret is only used during bootstrap, afterwards any other security mechanism can be used over the secure DTLS channel. Using asymmetric encryption would require more configuration.

On the other hand, Raw Public Key (RPK) mode also provides a good trade-off between resource consumption and security robustness. An advantage of using asymmetric encryption over PSK is that it enables the creation of identities based on the Public Key hash. Equally as in the PSK scenario, another cypher suite can be installed after Bootstrap. Raw Public Key with an out-of-band mechanism to validate the key provides the same security properties Certificate mode has. For this particular document we will be using Raw Public Key.

2 Actors

- LWM2M Client: A managed device running the client side of the LWM2M management protocol.
- LWM2M Bootstrap Server: A server tasked with providing the information to contact a LWM2M Server upon request from a client and after validation of the required credentials.
- LWM2M Server: A manager of a LWM2M Client, it can perform device management, service enablement and subscribe to management information from the device.
- Administrator: Manages the device through the LWM2M Server. It can be a human or an application.

- CoAP Server: A lightweight server running on the constrained device, it exposes a set of resources for applications.
- CoAP Client: It is a lightweight client that can access resources of a device running a CoAP Server.
- User: Can "use" the non-management information of the device with a CoAP Client. That is, access readings, actuate the device, subscribe to readings and other application-related operations. Applications can be a "users" of the device.

3 Device Pre-Provisioning

LWM2M's Object Model structure the information in Objects that can be instantiated, each of them containing a series resources with the actual data [3]. They are addressed with a number for the object, another for the instance and one last number for the resource: "object/objectID/resourceID"

We assume a typical scenario where a constrained device running a LWM2M Client needs to be Bootstrapped. A LWM2M Client contains a set of LWM2M Objects that define its properties. The set of mandatory objects and their respective Object IDs (OID) as defined by OMA are LWM2M Security (OID 0), LWM2M Server (OID 1), Access Control (OID 2), Device (OID 3), Connectivity Monitoring (OID 4), Firmware (OID 5), Location (OID 6), Connectivity Statistics (OID 7). Detailed information about the properties of each can be found in the Extension documents of the OMA LWM2M Specification [4].

Not all of those Object instances should be there already but a sensible assumption is that a device should have already information with its device model and manufacturer as well as bootstrapping information. Thus it should contain instances /0/0 and /3/0.

One "LWM2M Security Object" instance (/0/0) is provisioned on the LWM2M Client for the bootstrap server. Another instance (/0/1) about the LWM2M Server should be added during the bootstrap phase (Section 5) by the LWM2M Bootstrap Server. Each instance will contain the following resources:

- LWM2M Server URI: Uniquely identifies the LWM2M Server or LWM2M Bootstrap Server, and is in the form: "coaps://host:port".
- Bootstrap Server: Boolean value determining if the current instance concerns a LWM2M Bootstrap Server (true) or a standard LWM2M Server (false).
- Security Mode: Specifies which security mode it uses (PSK, Raw PK, etc).
- Public Key or Identity: In our case, as we are using RPK we would store the Clients public key. In other modes this would be used for the identity of the device.
- Server Public Key: Similarly, this stores the RPK of the server.
- Secret Key: If there is a secret key in the security mode, this is where it is stored.

There are also other attributes related to SMS binding and other optional parameters that we do not use.

Once the device is provisioned, we can initiate the bootstrapping process.

4 Other Pre-Provisioning

When using Raw Public Key (RPK), it is necessary to send the same bootstrap request, the hash of the RPK or some other identity by another conceptually different channel (independent from the main LWM2M data stream). We envision deployment as a quick task that should not take more than few seconds. Therefore we believe some form of barcode or QR code with the hash of the RPK could be sufficient to determine the public identity of the device. Such information has to be provisioned already on the device.

Additionally, the user scanning the QR code could do so with a dedicated application where he had previously logged-in (username/password), so that his identity is also valid in the system. This step is not required but it could give an extra degree of security during deployment since the QR code can be read by anyone who has physical access to the device.

The device should be able to get an IP address (IPv6 preferred) and should have connectivity towards the Bootstrap Server and LWM2M Servers, they all could be located on the same local network or the device could have Internet connectivity.

5 Bootstrapping and Registration of a device

For every CoAP REST request operation there will be a response. Possible CoAP methods are POST, PUT, GET and DELETE. Common responses will be "ACK 2.03 Valid" , "ACK 2.01 Created", "ACK 2.02 Deleted", more can be found on sections "12.1.1 Method Codes" and "12.1.2 Response Codes" of the CoAP Specification [1] and on section 8.5 of the LWM2M Specification [4]. The steps are the following:

1. The user installing the device activates the out-of-band public key validation mechanism. It scans the QR code that contains at least the hash of the RPK of the device. This is sent to the manager. This process can trigger a timer in which the system will expect a client-initiated bootstrap.

2. The user installing the device turns the device on. The device will switch on the antenna and try to get an IP address (we assume DHCP is also working), it is not required to use a secure Internet/Local connection but some form of connectivity is obviously needed too.

3. Once the device has IP connectivity it will try to establish a DTLS session with the LWM2M Bootstrap Server. The DTLS handshake is well-documented [2] [5]. The Bootstrap Server will use the informaiton sent through the out-of-band mechanism to validate the RPK of the device.

4. The LWM2M Client sends a "Request Bootstrap" to the LWM2M Bootstrap Server URI. It also sends its endpoint name as a URN identifier.

```
Req: POST  coaps://<ip>:<port>/bs?ep=<urn>
Res: 2.04 Changed
```

5. In this case, the next step is for the Bootstrap Server to initialize the device. This can be performed in multiple ways depending on the use case. In some cases you might to simply just add a new LWM2M Server, its security settings and associated ACL instances (/1/0 and /2/0):

   ```
   Req: PUT /0/1
   Res: 2.04 Changed
   Req: PUT /1/0
   Res: 2.04 Changed
   Req: PUT /2/0
   Res: 2.04 Changed
   ```

   In addition to adding a LWM2M Server you might also want to reset the previous bootstrapping information. Such step is optional however useful if we want to point to our own bootstrap server URI. This can be done in one or two steps, depending if we want to delete or simply overwrite:

   ```
   Req: DELETE /0/0
   Res: 2.02 Deleted
   ```

   and/or

   ```
   Req: PUT /0/0
   Res: 2.04 Changed
   Req: PUT /0/1
   Res: 2.04 Changed
   Req: PUT /1/0
   Res: 2.04 Changed
   Req: PUT /2/0
   Res: 2.04 Changed
   ```

6. The specification mandates a finish indication after it has sent all object instances/resources. Bootstrap Server send finish message by sending CoAP POST to "/bs" location path with empty payload

   ```
   Req: PUT /bs
   Res: 2.04 Changed
   ```

7. After this, the device now has the URI and key material to connect to the LWM2M Server, perform the DTLS handshake to the Manager and register the device. This is done by requesting to register the device at the Resource Directory (/rd) with ID ¡urn¿, lifetime 30000 seconds and UDP bindings.

   ```
   Req: POST  coaps://<ip>:<port>/rd?ep=<urn>&lt=30000&lwm2m=00&b=U
   Res: 2.01 Created Location: /rd/83j1
   ```

At this point the device is registered and can be managed. The following figure shows the previously explained scenario. The manager of the M2M devices is located in the Internet and runs at least one LWM2M Server; a bootstrapping service is also located in the Internet and runs a single LWM2M Bootstrap Server instance.

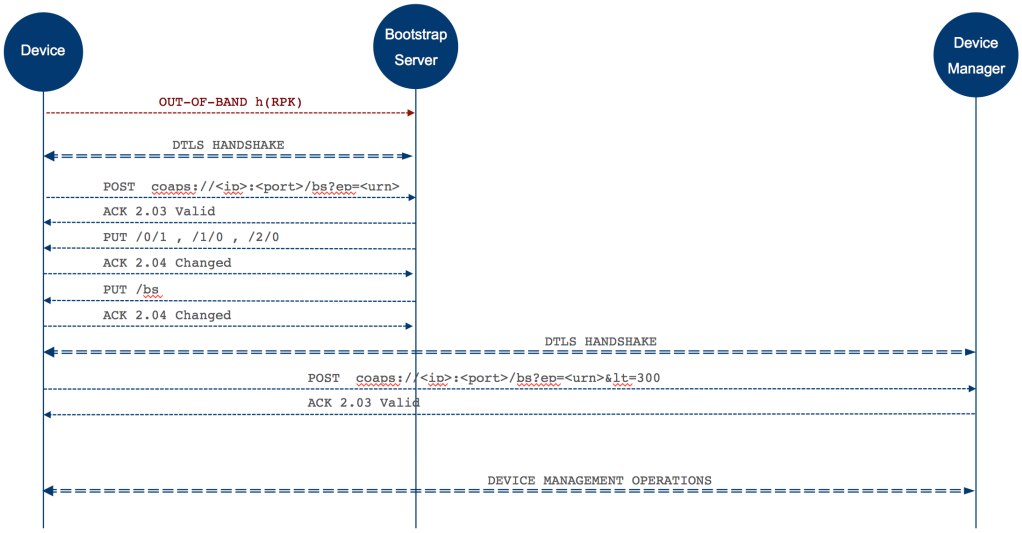

Figure 1: Bootstrapping and Registration of a Device

6 Using Applications

Once a device is bootstrapped we can perform any set of operations on it through the CoAP interface. Resources are exposed as Objects using the same data model used for management [3].

For example, in the case of a temperature sensor we can access and subscribe to the readings of the device:

```
Req: GET /3303/0/1 Observe_Option=1
Res: 2.05 Content (25 C)
Res (Notify): 2.05 Content (26 C)
```

We could turn off a buzzer to a controller:

```
Req: PUT /3338/0/5850 False
Res: 2.04 Changed
```

The Resource model is flexible enough to make it simple to create new resource representations whenever needed.

References

[1] CoAP Protocol. https://tools.ietf.org/html/rfc7252.

[2] DTLS. https://tools.ietf.org/html/rfc4347.

[3] IPSO Object Model. https://github.com/IPSO-Alliance.

[4] LWM2M Specification. http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m-v1-0.

[5] TLS. https://tools.ietf.org/html/rfc5246.