Heads-Up

SOLACE

Smart Object Lifecycle Architecture for Constrained Environments

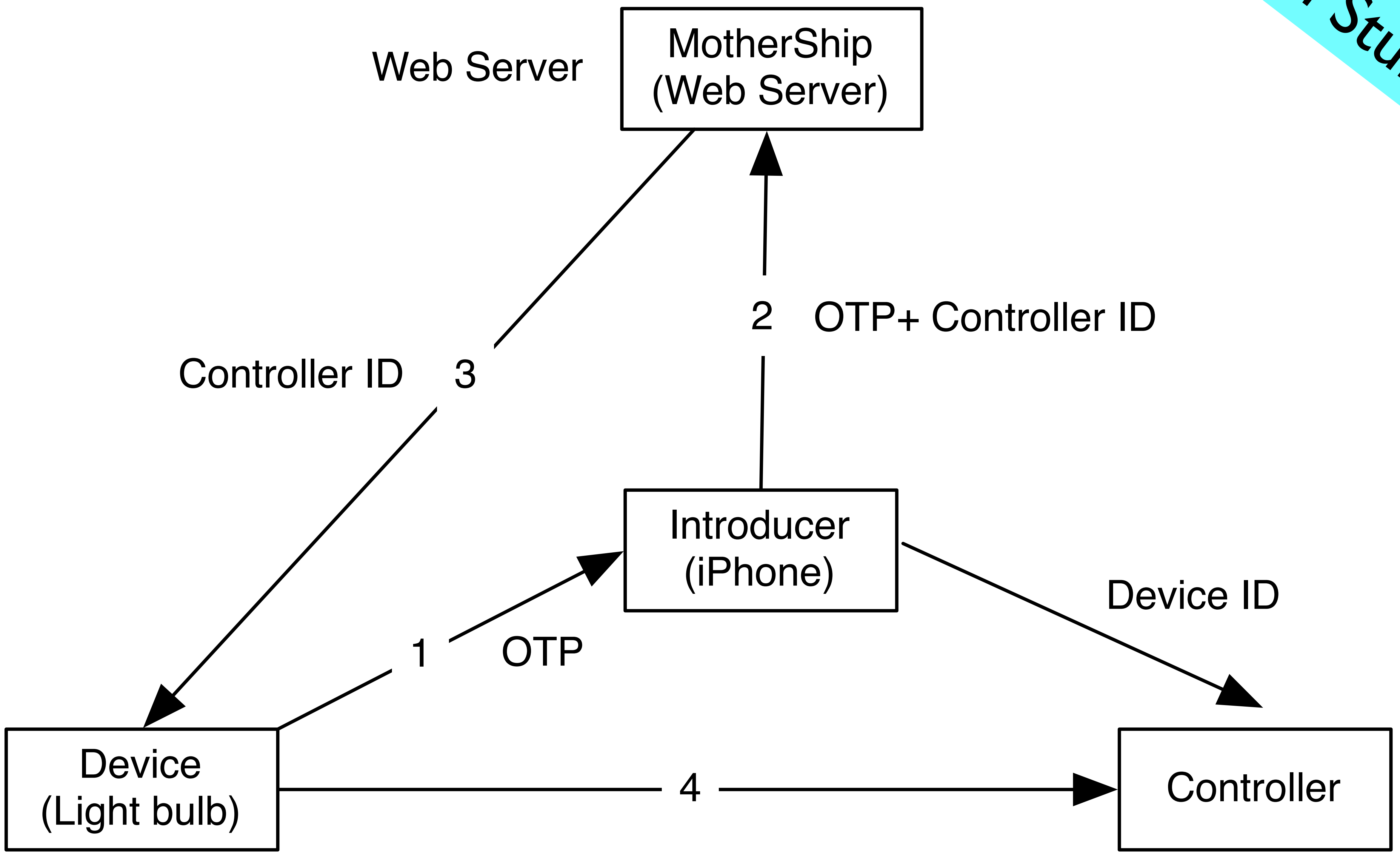
Where do I get my keys?

- IEEE 802.15.4 needs keys
- RPL needs keys
- CoAP/DTLS needs keys

- Lots of desire for
  key management protocols

Secure Bootstrapping Protocol

- We have a solution based on EAP-TLS and raw public keys as certificates
- Based on EAP authentication framework of RFC 5247 (covered in Annex C)
- EAP-TLS (RFC5216) certificate-based mutual authentication and key derivation protocol that uses TLS
- draft-ietf-tls-oob-pubkey extends TLS with raw public key support
- For CoAP devices the usage of X.509-based PKIX certificates is an unnecessary burden
- CoAP device can be configured with a client public key aka raw public key and use it as certificate
- Result: simplified authentication, no need for CAs, reduced code size


Web Server

MotherShip
(Web Server)

2 OTP+ Controller ID

Controller ID 3

Introducer
(iPhone)

Device ID

1 OTP

Device
(Light bulb)

4

Controller

Cool Stuff

draft-jennings-core-transitive-trust-enrollment-01.txt

Friday, March 23, 12

What do the keys do?

- Where can I use them?
- What do they authenticate? authorize?
- How do I re-key? get rid of their power?

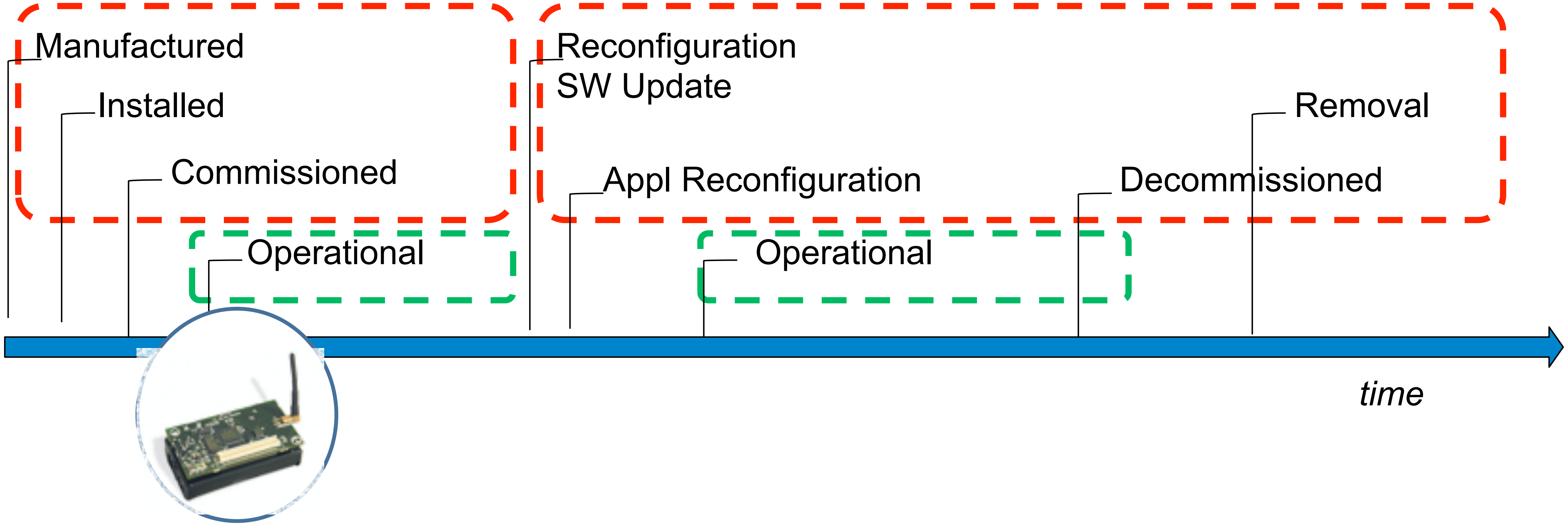
What are my security objectives, anyway?

- There is no security without security objectives
- Who tells us those? When? How?
- Who is authorized to make these decisions? Who did they authorize?
- Who owns stuff? data?

General security objectives

- Not subject to a mass attack
- Usable (yes,Virginia, that is a security objective)
- Channel security
- Authentication of participating entities
- Authorization of access to resources
- Maintains security over a **lifecycle**
- ...

Thing **lifecycle** and security framework



Manufactured

Installed

Commissioned

Operational

Reconfiguration
SW Update

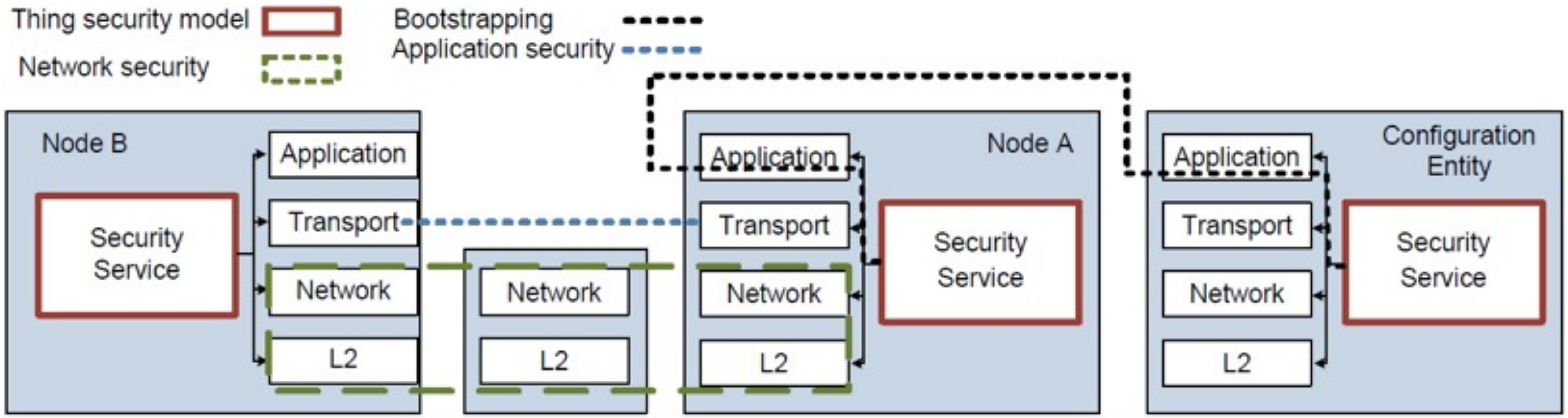
Appl Reconfiguration

Operational

Decommissioned

Removal

time



draft-garcia-core-security-04.txt

Objective

- Define enough of the **architecture** so:
  - we know what we are **talking about**
    - and have **terminology** for the components
  - we know when we have the **technology pieces** we need

Technology pieces

- **Cryptographic algorithms**: hash functions, keyed message digest, encryption functions, …
- **Enrollment**: leap of faith, PAKE, out-of-band provisioning, …
  - probably most relevant from **usability** p.o.v.
  - stay reasonable/**lightweight** per application
- Security **protocols**: TLS/DTLS, IKEv2, EAP-TLS, …
- **Credentials**: Raw Public Keys, PSK Identity, X.509 certificates, passwords, …

SOLACE: Where?

- We bounced it around IETF WGs for half a decade or so
- We got focused again in two **workshops**:
  - IAB Smart Object workshop **2011** http://tools.ietf.org/html/rfc6574
  - Smart Object Security workshop **2012** http://tools.ietf.org/html/draft-gilger-smart-object-security-workshop-00
- Where to do the work?
  - Start in the **IRTF**, and
    then do the missing pieces in the **IETF**
  - (Open for other approaches.)

SOLACE:
How to start it

- Define one (1) **usage scenario**/use case
- Solicit **contributions** that
  - **spec out** the smart object lifecycle,
    from manufacturing via initial keying, establishment of security associations, authorization, configuration, changes to all these (including re-keying), decommissioning (and de-authorization), and recycling/re-use.
  - considering network access, routing, and application layers
- Discuss and **extract** structure,
  elements of an architecture