Cryptography in AllJoyn, an Open Source Framework for IoT

Greg Zaverucha

Microsoft

Real World Cryptography Conference 2016

Internet of Things

Things are devices that have one or more sensors/functions and network connectivity

- Wearables (e.g., heart rate monitors)
- Industrial Sensors (e.g., Things on oil pipelines)
- Building automation (e.g., HVAC, CO2 detectors, etc.)
- Smart appliances (e.g., TVs, washing machines)
- Home automation (e.g., security system, lighting)

Marketing people call everything IoT

Lots of IoT-Related Technology

Multiple industry efforts to standardize protocols for "Things"

Multiple radios/transports

- 802.15.4, BTLE, WiFi, ZigBee, Zwave, 6lowpan

Protocols for discovery, routing, security

- AllJoyn, Thread, MQTT, IoTivity, CoAP

Multiple ecosystems

Protocol bridges

- Many scenarios require things to talk to each other
- E.g., thermostat using the home security system's motion sensors

Gateways

- Connectivity to the cloud

"Hub" model seems to be common

Lots of IoT-Related Technology

Multiple industry efforts to standardize protocols for "Things"

Multiple radios/transports

- 802.15.4, BTLE, WiFi, ZigBee, Zwave, 6lowpan

Protocols for discovery, routing, security

- AllJoyn, Thread, MQTT, IoTivity, CoAP

Multiple ecosystems

Protocol bridges

- Many scenarios require things to talk to each other
- E.g., thermostat using the home security system's motion sensors

Gateways

- Connectivity to the cloud

"Hub" model seems to be common

Outline

~~What is the Internet of Things (IoT)?~~

What is AllJoyn?

Overview of security features in AllJoyn

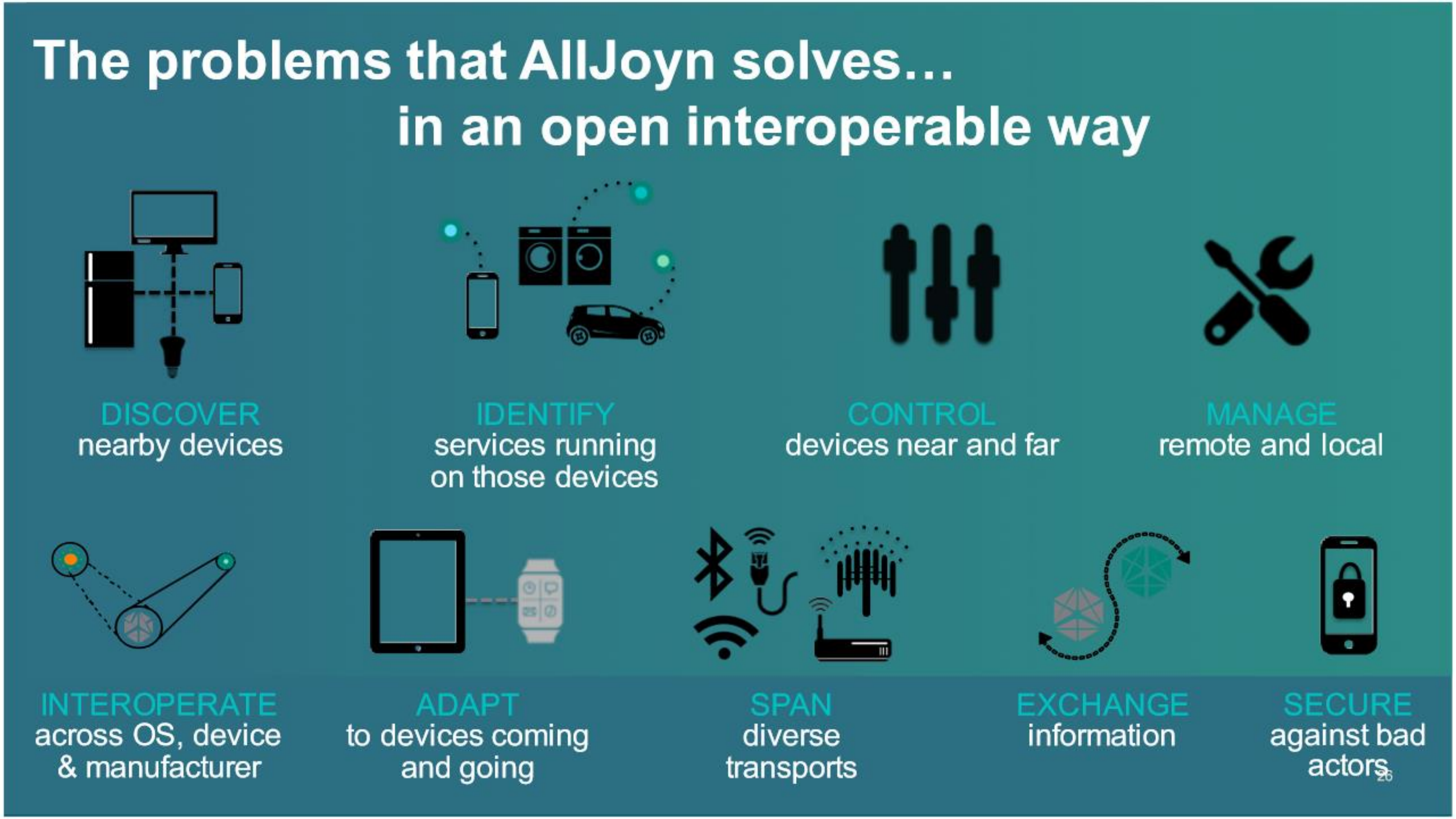
Details of secure channel establishment

Quick overview of device management features

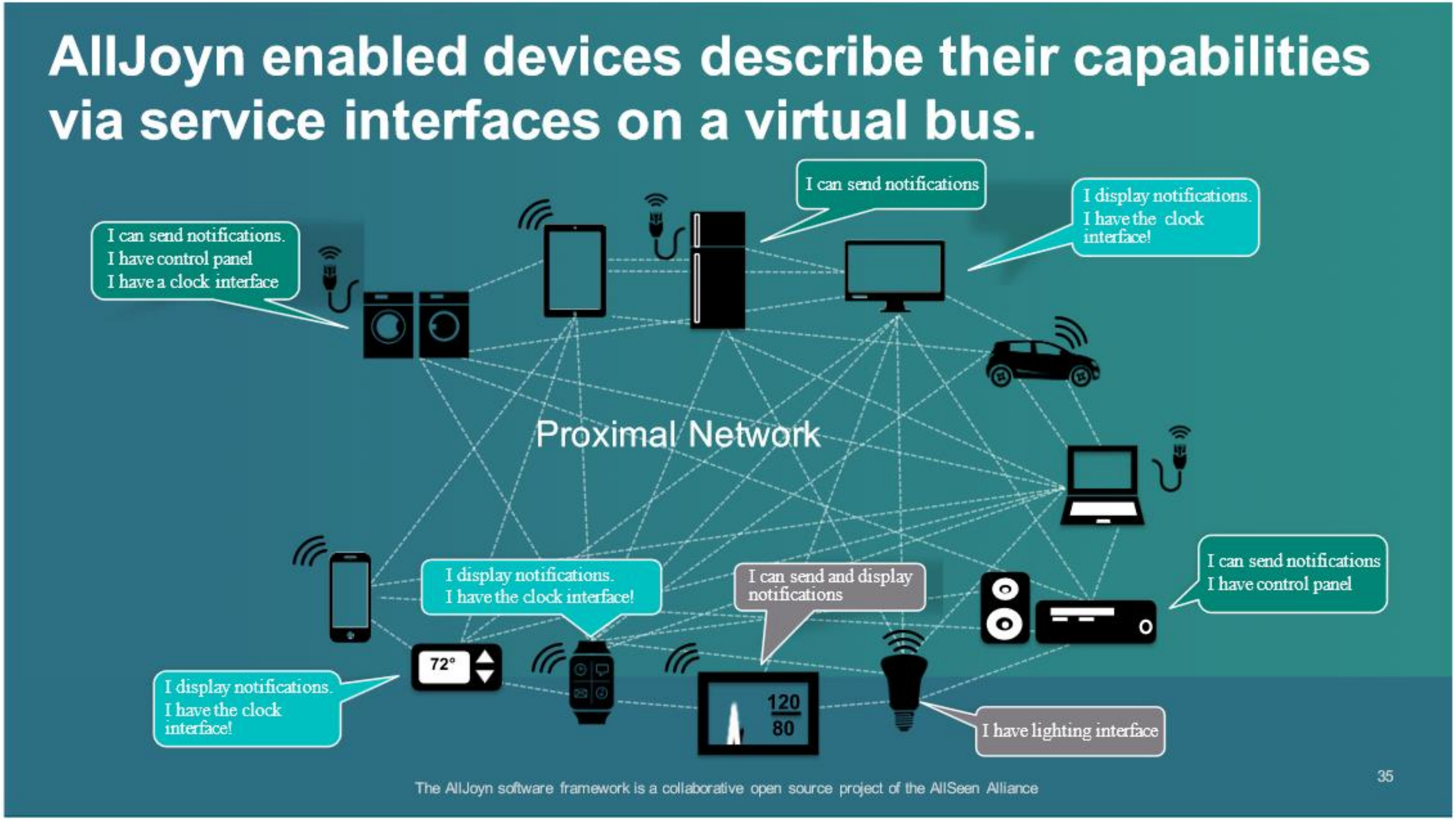
AllJoyn

Linux Foundation Collaborative Project AllSeen Alliance

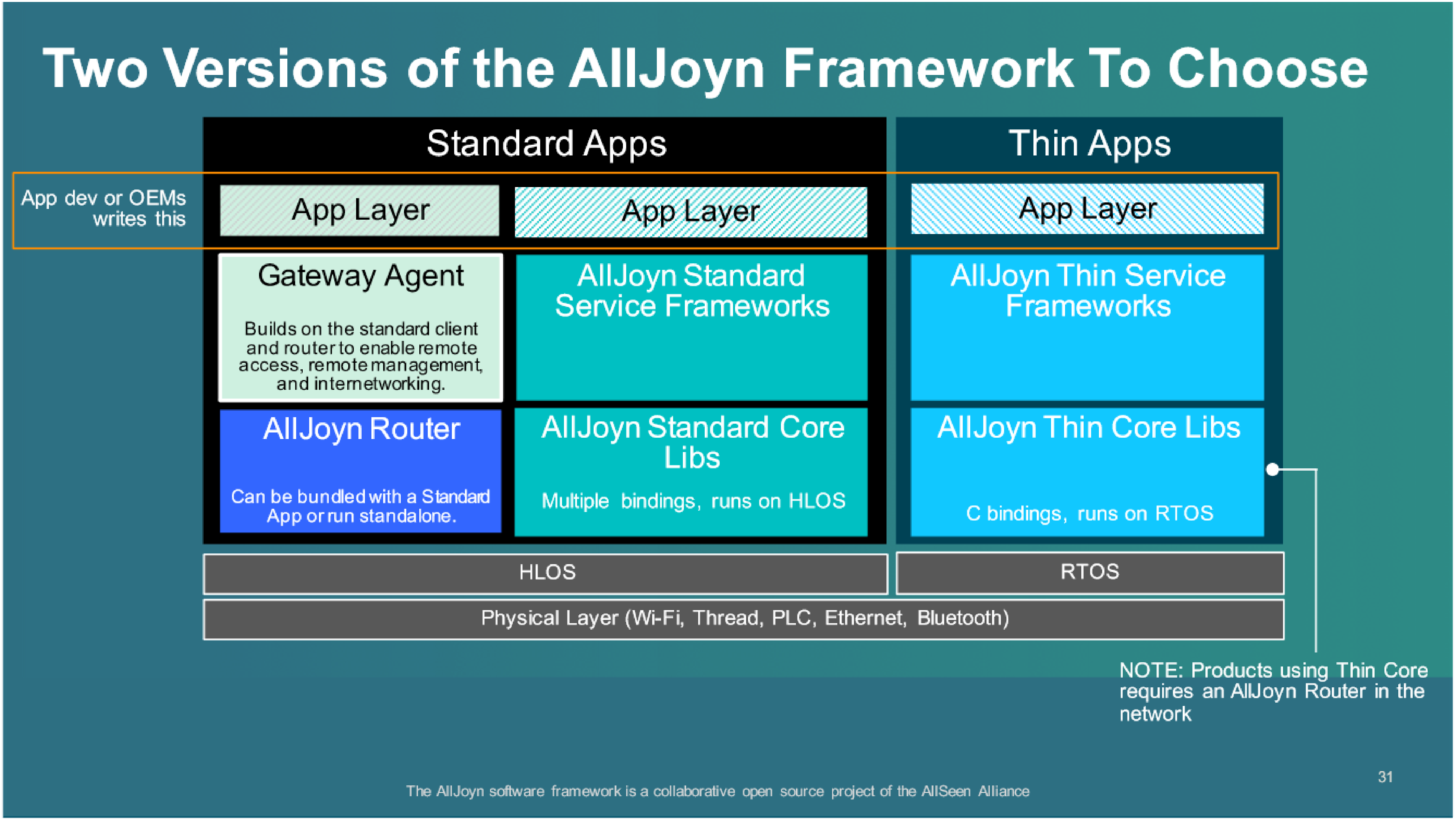
- Industry-wide open source effort
- 170 member companies
  - Microsoft, Qualcomm, Panasonic, Haier, LG, Sony, IBM, Cisco, Lenovo, AT&T, Netgear, Honeywell, D-Link, ADT, ZTE, HTC, Symantec, Vodafone, ASUS
- (Unofficial) focus on home automation & WiFi networks
- 10+ Microsoft employees involved, some here at RWC ☺
  - Kevin Kane (committer)
  - Dan Shumow (contributor)
  - Tim Ruffing (contributor, MS intern 2015)



Source: Overview of the AllSeen Alliance
https://allseenalliance.org/sites/default/files/resources/intro_to_alliance_9.4.15.pdf



Source: Overview of the AllSeen Alliance
https://allseenalliance.org/sites/default/files/resources/intro_to_alliance_9.4.15.pdf



Source: Overview of the AllSeen Alliance
https://allseenalliance.org/sites/default/files/resources/intro_to_alliance_9.4.15.pdf

AllJoyn Support in Windows 10

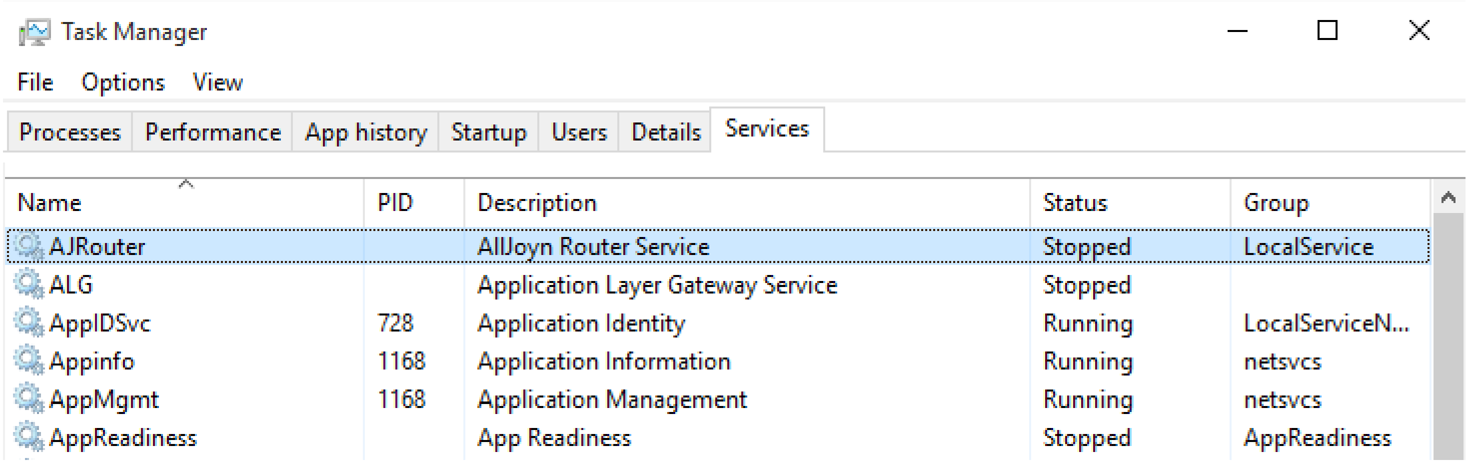
Built-in router

Windows API support

AllJoyn Studio plug-in for Visual studio

Code samples:

https://github.com/ms-iot



AllJoyn Security

AllJoyn Security Evolution

Security 1.0: AllJoyn framework can establish a secure channel. Apps must determine and manage trust relationships.

Security 2.0: AllJoyn supports trust domains (e.g., a household). AllJoyn can handle device provisioning and security management.

Threat Model

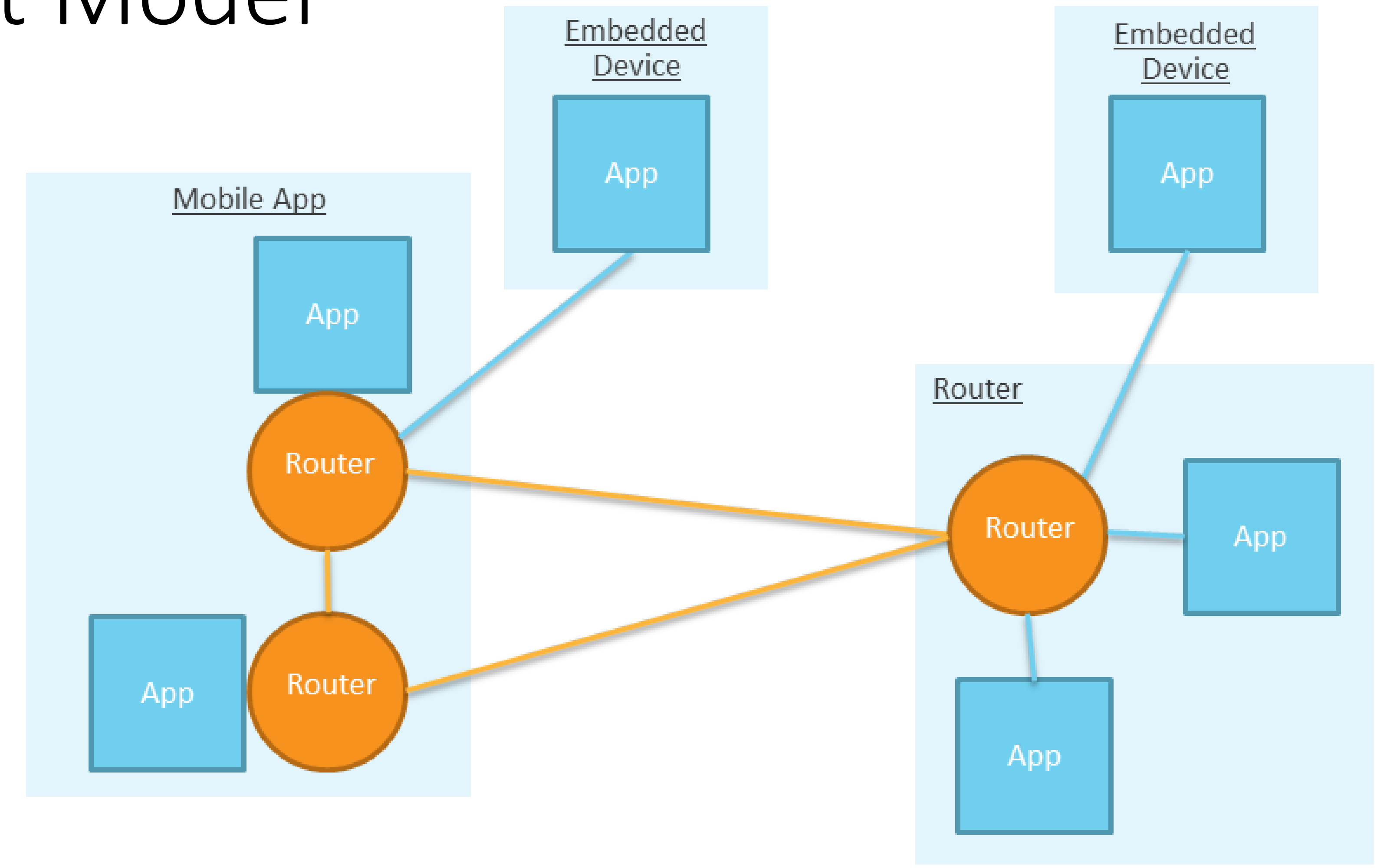


Image source: https://allseenalliance.org/sites/default/files/developers/learn

Threat Model

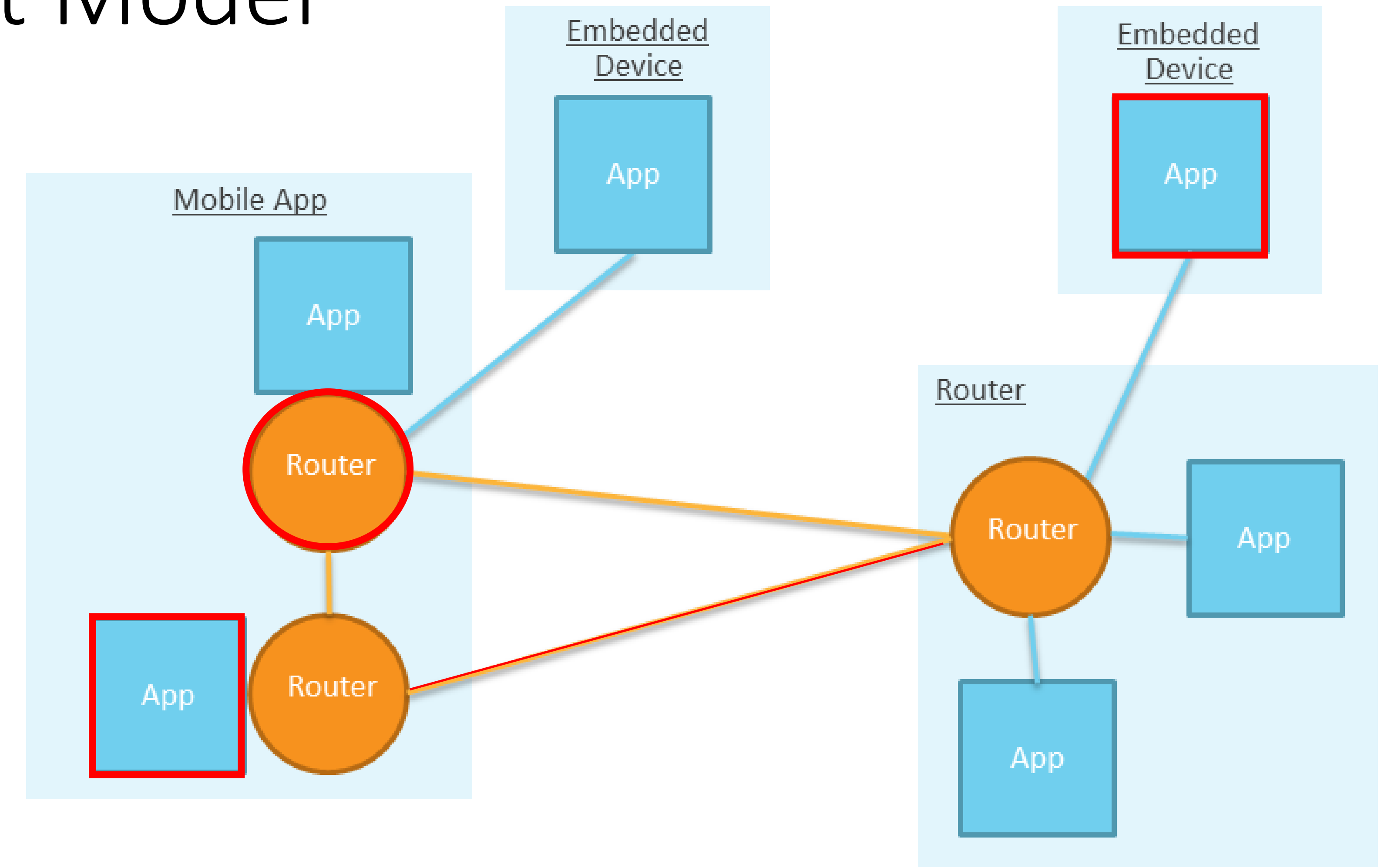


Image source: https://allseenalliance.org/sites/default/files/developers/learn

Threat Model

Attacker on the local network is able to interact with AllJoyn devices

- Can intercept and modify packets in transit (man-in-the-middle)
- Can drop and replay packets
- Can compromise some of the AllJoyn devices on the network

Examples

- Malware on the WiFi access point
- Malicious smartphone application
- Malicious device on the network

Attackers could be physically nearby or remote

Security goal is secure channel establishment

(D)TLS?

- AllJoyn design is intended to be transport agnostic
  - Protocol is defined in terms of messages
  - Transport is not necessarily IP (e.g., Bluetooth)
  - Having security above the transport layer ensures equal security regardless of transport
- TLS could probably be used with TCP transport option
  - And DTLS with UDP
  - With significant cost in terms of development and compatibility
- AllJoyn security protocols are derived from TLS, similar
  - But with far fewer options/extensions

Key Exchange Authentication Mechanisms

ECDHE: Elliptic Curve Diffie-Hellman (Ephemeral)

- Fresh key pair generated for each exchange
- Long term credential used for authentication only
- Always mutual authentication

Multiple ways to authenticate key exchange

- NULL: no authentication. Vulnerable to active MITM attacks
- PSK: authentication by pre-shared key (PSK). Secure if PSK has high entropy
- ECSPEKE: password-based authentication. To be added in 16.04 release
- ECDSA: authenticated with an ECDSA signature. Certificates exchanged and validated

Key Exchange Authentication Mechanisms

Security 1.0 provides all options to apps, they decide which mechanisms to support, and which to require

Security 2.0 uses only ECDHE_ECDSA after setup

EC-SPEKE will replace PSK as the preferred way to secure setup

- Easier to use (password vs. PSK entropy)
- The protocol is a profile of SPEKE from IEEE 1363.2
- Protocol-wise, almost as simple as replacing the base point in ECDHE_NULL
- Design document on Core WG wiki (wiki.allseenalliance.org)

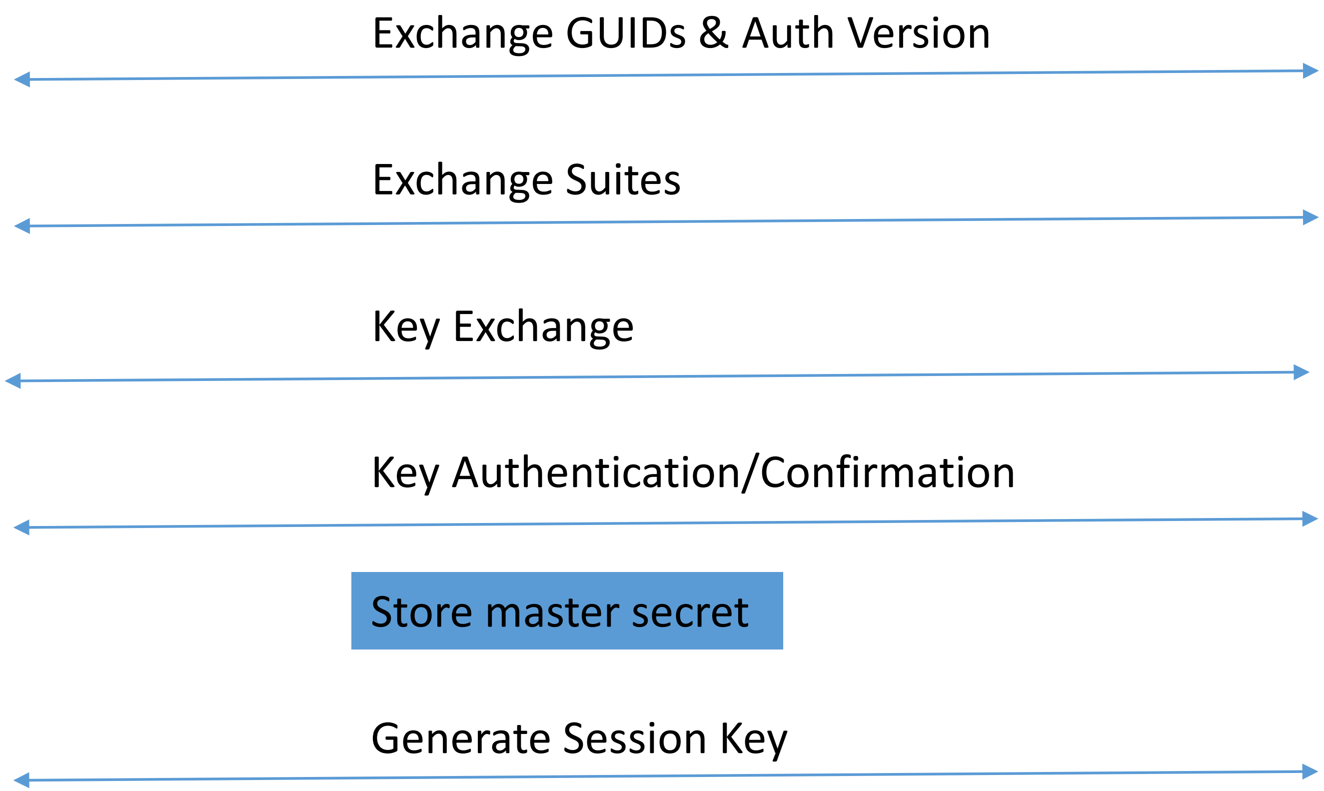
Parameters and Algorithms

Algorithms and parameters are fixed per authentication version

Primitives are all from existing standards, 128-bit security level

- Key exchange: ECDH (SP800-56A)
- Signatures: ECDSA (FIPS186-4)
- Curve parameters: NIST P256 (FIPS186-4)
- Data encryption & authentication: AES CCM
- Hashing: SHA-256
- Key derivation: the “TLS PRF” from RFC 5246
- Certificates are X.509 (RFC 5280) + AllJoyn EKUs and extension

AllJoyn Key Exchange Overview



Session Resumption

Exchange GUIDs & Auth Version

~~Exchange Suites~~

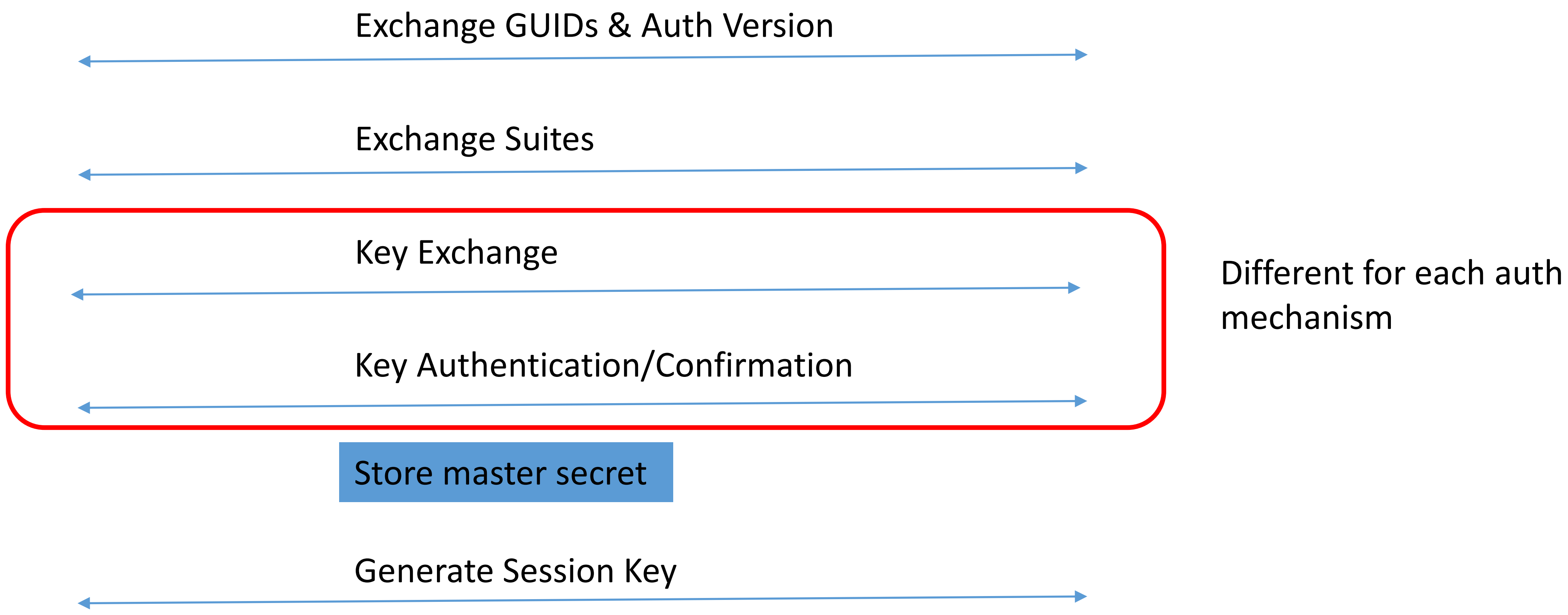
~~Key Exchange~~

~~Key Authentication/Confirmation~~

Retrieve stored master secret

Generate Session Key

AllJoyn Key Exchange Overview



ECDHE_ECDSA Key Exchange

Exchange GUIDs, Auth Version, Auth Suites

⋮

Key Exchange

Generate (Q_A, s_A)

Q_A →

Generate (Q_B, s_B)
Compute $z = \text{ECDH}(Q_A, s_B)$
Compute $M_B = \text{PRF}(\text{SHA-256}(z), \text{"master secret"})$

← Q_B

Compute $z = \text{ECDH}(Q_B, s_A)$
Compute $M_A = \text{PRF}(\text{SHA-256}(z), \text{"master secret"})$

ECDHE_ECDSA Key Authentication

Exchange GUIDs, Auth Version, Auth Suites, Key Exchange

⋮

Key Authentication

h_A :=SHA-256(all msgs)
L := “server finished”
V_A = PRF(M_A, h_A, L)
Sig_A = ECDSASign(…, V_A)

$Sig_A, Cert_A$ →

Validate $Cert_A$
h_B := SHA-256(all msgs)
Re-compute V_A using M_B and h_B
ECDSAVerify($Cert_A, Sig_A, V_A$)
L := “client finished”
V_B = PRF(M_B, h_B, L)
ECDSASign(…, V_B)
Store M_B

← $Sig_B, Cert_B$

Validate $Cert_B$
Re-compute V_B using M_A and h_A
ECDSAVerify($Cert_B, Sig_B, V_B$)
Store M_A

ECDHE_ECDSA Generate Session Key

Exchange GUIDs, Auth Version, Auth Suites, Key Exchange, Key Authentication

⋮

Generate Session Key

Choose nonce N_A

N_A →

Choose nonce N_B

$K_{BA}||V_B$:= PRF(M_B, $N_A||N_B||$"session key")

← N_B, V_B

$K_{AB}||V_B$' := PRF(M_A, $N_A||N_B||$"session key")

Ensure V_B == V_B'

Start using K_{AB}

Security 2.0 Overview

Trust Model Changes

With Security 1.0, apps were responsible for

- Provisioning credentials
- Establishing trust with other apps
- Implementing access control on certain interfaces, if required

Doesn’t scale to the household scenario

- Devices made by different manufacturers
- More than one user, guest access, …

Security 2.0 adds a *security manager*, per trust domain

- E.g., one per household

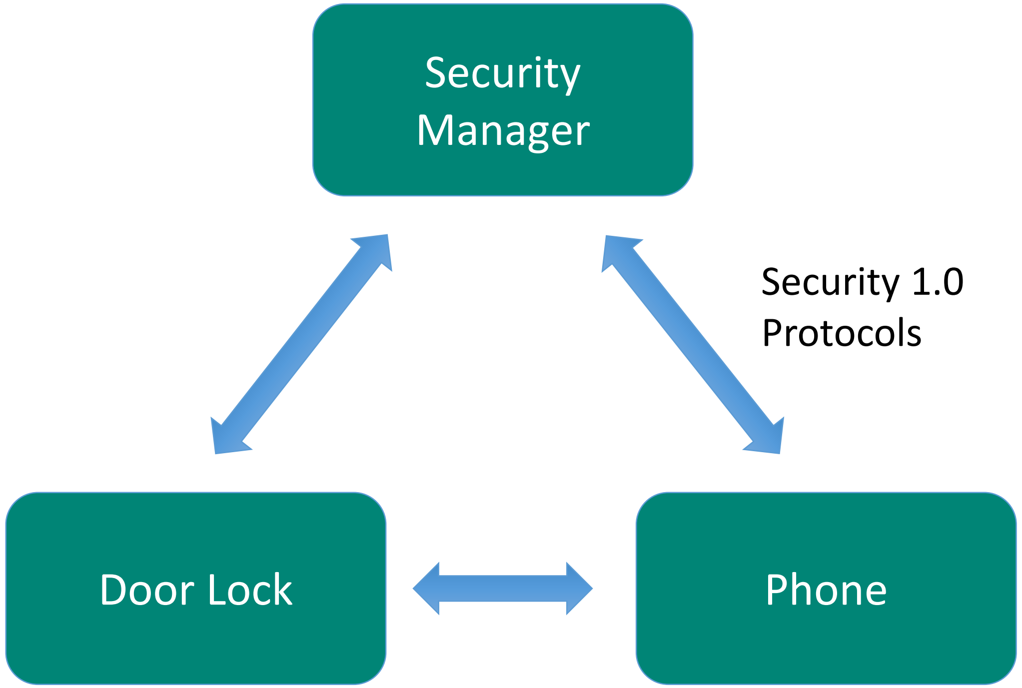
Security 2.0 Overview

New AllJoyn devices/apps are in "claimable" state when they join the network

The security manager claims them and provisions certificates and policy

Certificates are used for identity and membership in security groups

Bootstrapping only required between security manager and apps



Security 2.0: Policy

Apps that produce interfaces have access control policies

- Interface and method level granularity
- Can refer to security groups or individual apps

E.g., only allow members of the ADMIN group to access the PinCodeChange interface on the door

E.g., only allow Alice and Bob’s phones to open the garage door

Security 2.0: Manifests

Manifests: apps list the interfaces they consume, the list is approved and certified by the security manager, then enforced by producers.

- Failed manifest check will deny access even if allowed by policy
- Similar to mobile apps requesting API access

E.g., A lighting control panel app's manifest lists lighting interfaces. The alarm system will deny access to the motion sensor interfaces.

Links and resources

- Security 2.0 documentation:
  - https://allseenalliance.org/framework/documentation/learn/core/security2_0/hld
- Source code
  - https://git.allseenalliance.org/cgit/
  - alljoyn.git and ajtcl.git are the standard and thin client implementations
- Mailing lists
  - https://lists.allseenalliance.org
  - allseen-core, allseen-security are most relevant
- General AllJoyn info
  - https://allseenalliance.org/framework
- Windows AllJoyn API documentation
  - https://msdn.microsoft.com/en-us/library/windows/desktop/mt270094%28v=vs.85%29.aspx