Cryptography in AllJoyn

Greg Zaverucha
Software Engineer, Microsoft
Agenda

1. Review of AllJoyn security features
2. Authentication and security protocols
3. Comparison to Internet protocols
4. Leveraging AllJoyn crypto

Talking about versions 15.04, 15.09
Motivation

This talk will not argue that AllJoyn/IoT needs security – given.

Why is AllJoyn crypto important?
It’s necessary for security.
Should be the “strongest link”.
Review of AllJoyn Security Features
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 choose to telecommute
Security Goals

• Authentication/Integrity; apps know who they are talking to and that messages are received as sent
• Confidentiality; message contents are kept private
• DoS resistance; attacker should not be able to easily deny service
• Perfect forward secrecy; compromise of an endpoint does not compromise past sessions
• Security 1.0 Example
• Security 2.0 Example
Security 1.0 Example

• Bootstrap authentication with a shared credential: password, key
  – Alternatively: Trust a certificate on first use, then pin it

• Credential type will dictate the authentication mechanism that AllJoyn will use

• Once a master secret is established, session keys are derived from it
Security 2.0 Example

- New AllJoyn devices are in “claimable” state when they join the network
- The security manager claims them and provisions certificates and policies
- Certificates are used for identity and membership in security groups
- Bootstrapping only required between security manager and apps
Authentication Mechanisms and Crypto Algorithms
Authentication and Key Exchange Protocol

Establish a secure channel in three main steps

1. Exchange GUIDs – Do we know each other?
   If so, do we share a master secret? (that hasn’t expired)

2. [Key Exchange]
   Authenticate the peer and establish the master secret
   Store master secret in the key store

3. Derive Session key
   Derive a session key from nonces and master secret
Key Exchange

- **ECDHE**: Elliptic Curve Diffie-Hellman (Ephemeral)
  - Long term credential used for authentication only
  - Fresh key pair generated for each exchange

- **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 (128 bits or greater)
  - **ECDSA**: authenticated with an ECDSA signature. Certificates exchanged and validated

- **SRP (SC only, deprecated)**
  - Similar use cases as ECHDE_PSK, but secure with low-entropy secrets
Key Exchange Crypto Algorithms

AllJoyn uses standard algorithms and parameters

- **ECDHE**
  - As specified in SP800-56A

- **ECDSA**
  - As specified in FIPS 186-4, ANSI X9.62
  - Only signature algorithm in key exchange protocol and certificates

- **Elliptic curve parameters**
  - EC parameters are not negotiated
  - Fixed to one set, the 256-bit NIST curve “P256”
  - Balance between performance and security
Certificates

- Standard X.509 v3 certificates (RFC 5280)
  - Signature algorithm is always ECDSA
  - Curve parameters are always P256
  - Hash algorithm is always SHA-256
  - Supports PEM encoded certificates created with common tools

- AllJoyn specific EKU fields indicate the certificate type
  - In Security 2.0 usage, there are identity and membership types

- Chains typically have length two:
  - The security manager is the issuer/root and the app is the end-entity

- Revocation is not implemented
  - Security manager can use policy to deny access
Encryption of Data in Transit

- Once a session key is established, data is encrypted and authenticated

- Data is protected with is AES in CCM mode (Counter with CBC-MAC, RFC 3610)
  - Authenticated encryption mode
  - Uses one key, 128-bits long, authentication tags are 128-bits as well
  - 112-bit IV: 32-bit counter + 96 random bits
Hashing and Key Derivation

• All hashing is done with SHA-256
  – With the exception of SRP, which uses SHA-1

• Signing & certificates

• Conversation hash
  – A hash of all authentication protocol messages ensures integrity of the authentication conversation

• Key derivation is done using the “TLS PRF” (RFC 5246, Section 5)
Security Level

• Overall the collection of AllJoyn crypto algorithms is expected to meet the 128-bit security level
  – 128-bit encryption keys and authentication tags
  – 256-bit hash digests and elliptic curve parameters
  – Assuming random number generation has the same security level

• Complexity of attacks should be around $2^{128}$ operations
  – NIST (SP800-57) and ANSSI guidance: good until 2030 and beyond
  – ECRYPT: 2015 to 2040
  – BSI: 2017-2021 at least (no guidance beyond 2021)
  – Summary of recommendations at www.keylength.com
Comparison to TLS and the Web
## Differences

<table>
<thead>
<tr>
<th>AllJoyn</th>
<th>TLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed ciphersuite</td>
<td>Negotiate ciphersuite</td>
</tr>
<tr>
<td>One set of curve parameters</td>
<td>Negotiated</td>
</tr>
<tr>
<td>One certificate signature</td>
<td>Flexible. RSA signatures are most common</td>
</tr>
<tr>
<td>algorithm, params.</td>
<td></td>
</tr>
<tr>
<td>No protocol extensions</td>
<td>Well-defined extension mech.</td>
</tr>
<tr>
<td>New, less security review</td>
<td>Highly scrutinized</td>
</tr>
</tbody>
</table>
## Differences (con’t)

<table>
<thead>
<tr>
<th>AllJoyn</th>
<th>TLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static relationships with fewer peers</td>
<td>Variable duration, diverse peers</td>
</tr>
<tr>
<td>Typically a local PKI</td>
<td>Typically global PKI (web)</td>
</tr>
<tr>
<td>Mutual auth. always</td>
<td>Typically server auth. only</td>
</tr>
<tr>
<td>Renegotiation used a lot</td>
<td>Less use of renegotiation</td>
</tr>
</tbody>
</table>
Similarities

Many crypto-level similarities with TLS (as commonly deployed)
Data from SSLLabs.com (September 2015)

- ECDHE use with TLS on the rise
  - About 70% of TLS servers support ciphersuites with PFS
- Use of SHA-256 for certificates preferred in TLS (71%)
- Authenticated encryption modes preferred in TLS 1.3
  - More AES use now that RC4 is being turned off
- PRF is the same
- Both use X.509 certificates
- AllJoyn’s use of a master secret is similar to renegotiation in TLS
Leveraging AllJoyn Crypto
Crypto Considerations when Using AllJoyn Security

- AllJoyn apps shouldn’t need to design security protocols
- Leverage core protocols to simplify application security design
- Security 1.0 apps must
  - Choose the authentication mechanisms to support
  - Decide how to bootstrap authentication and provision credentials
- Security 2.0 apps must
  - Have a security manager to work with
  - Choose between NULL and PSK auth. for first contact with security manager
    - Security manager should probably accept both
    - Apps (esp. devices) should prefer PSK
Certificate Validation

• Security 1.0 applications must validate certificate chains with a callback
  – The AllJoyn core doesn’t know who the app trusts
  – Requires care to do well
  – Some API support and sample code is available

• Security 2.0 uses certificates after bootstrapping and has enough information to validate certificates
  – The security manager installs the necessary root certificates
  – Applications don’t have to worry about cert validation
Secure Key Storage

- The core library keeps cryptographic keys in a keystore.
- By default it lacks strong protections, since these are platform specific, e.g.,
  - Available hardware for secure storage differs across devices
  - An app may rely on the platform to isolate its storage from other apps on the system
- Especially important to protect the security manager keystore in Security 2.0 deployments
  - Security manager root key can add devices and policies to the network and grant access arbitrarily
Crypto Implementation Details

• Standard client (SC) apps shouldn’t have to implement any crypto
  – Authenticated, secure channels can be established with the core library
  – Leverages platform crypto where possible (CNG and OpenSSL)
  – Also has a “built-in” option that has no dependencies
Crypto Implementation Details

• Thin Client (TC) apps/devices may want to make optimizations
  – TC includes all crypto. Not optimized, generic C
  – Devices may offload to hardware, or have platform-specific ASM
  – Benchmark **first**, consider scenario
  – Small amounts of platform-specific ASM will go a long way
  – Make sure to review the slides of yesterday’s TC Porting talk by Mathew Martineau and Peter Krystad
Crypto Implementation Details

• Crypto needs a source of entropy (important on TC devices)
  – DRBG implementation needs a seed

• Built-in implementations have protections for common side-channels attacks
  – Execution time and memory access pattern are independent of secret values
Summary

• AllJoyn includes cryptographic primitives for authentication and secure communications
  – Well understood, conservative algorithms, 128-bit security

• AllJoyn applications should leverage these mechanisms so their developers can focus on other issues
  – Some attention required to use the primitives correctly

• Feedback is welcome
  – Is something missing?
  – Deployment, porting pain points?
  – Is performance acceptable?
Resources and Links

• Related AllSeen Summit events
  – Wed 1 pm: Security 2.0 Overview, Dave Thaler (Microsoft)
  – Wed 3 pm: Security BoF Meeting
  – Multiple other security-related talks on the program

• Security 2.0 HLD

• Source code
  – core/alljoyn
  – core/ajtcl

• Core WG mailing list