AllJoyn™ Security 2.0 Feature

High-Level Design Document

Draft Rev 1 Update 6

March 11, 2015

Contents

1 Introduction 5

1.1 Purpose and scope 5

1.2 Revision history 5

1.3 Acronyms and terms 5

2 System Design 8

2.1 Overview 8

2.2 Premises 9

2.3 Typical operations 11

2.3.1 Claim a factory-reset application 11

2.3.2 Define a security group 15

2.3.3 Example of building a policy 15

2.3.4 Install a policy 15

2.3.5 Add an application to a security group 16

2.3.6 Add a user to a security group 16

2.3.7 Delegating membership certificate 17

2.3.8 Add a security group equivalence instruction to an application 19

2.3.9 Certificate revocation (not fully designed) 20

2.3.10 Distribution of policy updates and membership certificates (not fully designed) 20

2.3.11 Application Manifest (discussion-in-progress) 21

2.4 Access validation 23

2.4.1 Validating a producer policy 23

2.4.2 Validating a consumer policy 23

2.4.3 Exchanging membership certificates during session establishment 24

2.4.4 Anonymous session 25

2.4.5 Validating an admin user 27

2.4.6 Emitting a session-based signal 27

2.5 Authorization data format 28

2.5.1 The format is binary and exchanged between peers using AllJoyn marshalling 28

2.5.2 Format Structure 29

2.5.3 Policy Templates 33

2.6 Certificates 33

2.6.1 Main Certificate Structure 33

2.6.2 Identity certificate 34

2.6.3 Membership certificate 35

2.7 Sample use cases 35

2.7.1 Users and devices 35

2.7.2 Users set up by Dad 37

2.7.3 Living room set up by Dad 38

2.7.4 Son's bedroom set up by son 39

2.7.5 Master bedroom set up by Dad 40

2.7.6 Son can control different TVs in the house 41

2.7.7 Living room tablet controls TVs in the house 42

3 Enhancements to Existing Framework 43

3.1 Crypto Agility Exchange 43

3.2 Permission NotifyConfig Announcement 44

4 Future Considerations 45

4.1 Broadcast signals and multipoint sessions 45

Figures

Figure 2‑1. Security system diagram 9

Figure 2‑2. Claim a factory-reset application without out-of-band registration data 12

Figure 2‑3: Claim a factory-reset application without using out-of-band registration data and install admin security group 13

Figure 2‑4. Claiming a factory-reset application using out-of-band registration data 14

Figure 2‑5. Install a policy 15

Figure 2‑6. Add an application to a security group 16

Figure 2‑7. Add a user to a security group 17

Figure 2‑8. Distribution of policy update and certificates 21

Figure 2‑9: Building Policy using manifest 22

Figure 2‑10. Validating a producer policy 23

Figure 2‑11. Validating a consumer policy 24

Figure 2‑12. Exchange membership certificates 25

Figure 2‑13. Anonymous access 26

Figure 2‑14. Validating an admin user 27

Figure 2‑15. Validating a session-based signal 28

Figure 2‑16: Authorization Data Format Structure 29

Figure 2‑17. Use case - users set up by Dad 37

Figure 2‑18. Use case - living room set up by Dad 38

Figure 2‑19. Use case - son's bedroom set up by son 39

Figure 2‑20. Use case - master bedroom set up by Dad 40

Figure 2‑21. Use case – Son can control different TVs in the house 41

Figure 2‑22. Use case - Living room tablet controls TVs 42

Tables

Table 2‑1. Security 2.0 premises 9

Table 2-2: Action Mask Matrix 31

# Introduction

## Purpose and scope

This document captures the system level design for the enhancements to the AllJoyn™ framework to support the Security 2.0 feature requirements. Related interfaces and API design is captured at a functional level. Actual definition for interfaces and APIs is outside the scope of this document. Features and functions are subject to change without notice.

## Revision history

|  |  |  |
| --- | --- | --- |
| Revision | Date | Change Log |
| Rev 1 Update 0 | August 8, 2014 | Update with new format and comments |
| Rev 1 Update 1 | August 27, 2014 | Update with comments from the collaboration meeting |
| Rev 1 Update 2 | September 8, 2014 | Update with comments and agreement from the technical conference call on September 3, 2014. |
| Rev 1 Update 3 | October 30, 2014 | Update the authorization data section based on agreement from the technical conference call on October 14, 2014. |
| Rev 1 Update 4 | December 23, 2014 | Update the Certificate section and changes listed in JIRA tickets ASACORE-1170, 1256, 1259, 1260. |
| Rev 1 Update 5 | January 15, 2015 | Update the rule enforcing table after the conference call on Janurary 13, 2015 by the Security2.0 working group. |
| Rev 1 Update 6 | March 10, 2015 | Update the authorization data after the conference call on Janurary 20, 2015 by the Security2.0 working group. Updated the permission matrix to reflect the concept of Provide permission.  Updated based on review comments by the Security 2.0 working group on March 6, 2015. |

## Acronyms and terms

| Acronym/term | Description |
| --- | --- |
| About data | Data from the About feature. For more information, refer to the [About Feature Interface Spec](https://allseenalliance.org/docs-and-downloads/documentation/alljoyn-about-feature-10-interface-specification). |
| ACL | Access Control List |
| AES CCM | The Advanced Encryption Standard 128-bit block cypher using Counter with CBC-MAC mode. Refer to [RFC 3610](http://tools.ietf.org/html/rfc3610) for more information. |
| Producer | An AllJoyn application providing services on the AllJoyn network. |
| Consumer | An AllJoyn application consuming services on the AllJoyn network. |
| Device | A physical device that may contain one or more AllJoyn applications. In this document, whenever the term “device” is used, it indicates the system application of the given physical device. |
| AllJoyn framework | Open source peer-to-peer framework that allows for abstraction of low-level network concepts and APIs. |
| DSA | Digital Signature Algorithm |
| ECC | Elliptic Curve Cryptography |
| ECDHE | Elliptic Curve Diffie-Hellman Ephemeral key exchange |
| ECDHE\_ECDSA | ECDHE key agreement with asymmetric DSA based authentication. |
| ECDHE\_NULL | ECDHE key agreement only. No authentication. |
| ECDHE\_PSK | ECDHE key agreement with symmetric key/pin/password based authentication. |
| User | The person or business entity interacting with AllJoyn applications. |
| Factory-reset device | A device is restored to the original configuration. |
| Friend | A user who has a trusted relationship with the owner |
| Grantee | The application or user who is the subject of a certificate. |
| GUID | Globally Unique Identifier. A 128 bit identifier generated randomly in a way that the probability of collision is negligible. |
| Security Group | A logical grouping of devices, applications, and users. It is identified by a group ID which is a GUID and the group authority public key. An application can be installed with a policy to expose services to members of the security group. An application or user holding a membership certificate is in fact a member of the security group. Any member of the security group can access the services exposed to the group by the applications with ACLs defined for that group. |
| Security Group Authority | A security group authority is the user or application that defines the security group and grant membership certificates to other. The security group authority is the certificate authority for that group. |
| Holder | The application or user possessing a certificate. |
| OOB | Out Of Band |
| Permission Management module | The AllJoyn Core module that handles all the permission authorization. |
| PermissionMgmt | A set of AllJoyn interfaces to manage the permissions for the AllJoyn application. The implementation is provided by the Permission Management module |
| Security Manager | An application used to manage cryptographic keys, and generate and distribute certificates. |
| Security Appliance | A security appliance is a type of Security Manager that is always on. |
| Peer | Application participating in the AllJoyn messaging. |
| SHA-256 | Secure Hash Algorithm SHA-2 with digest size of 256 bits or 32 bytes. |
| Trust profile | Information used by peers to introduce themselves when contacting each other. |
| Certificate Authority (CA) | Entity that issues a digital certificate |
| Keystore | A repository of security keys and certificates. An application has one keystore. Multiple applications can share the same keystore, but they are considered as the same application using the same security principal. |
| Authentication GUID | The Authentication GUID is a GUID assigned to an application for authentication purpose. This GUID is persisted in the key store and provides a long-term identity for the application. Typically, this GUID is associated with a single application. In the scenario where a group of related applications share a given key store, they also share the same authentication GUID.  This GUID is used as a mapping key for storing and accessing authentication and encryption keys. All key materials associated with another peer is stored in the key store with the peer’s authentication GUID as the mapping key. |

# System Design

## Overview

The goal of the Security 2.0 feature is to allow an application to validate access to interfaces or secure objects based on policies installed by the owner. This feature is part of the AllJoyn Core library. It is not an option for the application to enforce permission. It is up to the user to dictate how the application performs based on the access control lists (ACLs) defined for the application. The AllJoyn Core Permission Management component does all the enforcement including the concept of mutual or one-way authorization before any message action can be taken.

The Security Manager is a service that helps the user with key management and permission rules building. Using policy templates defined by an application developer, the Security Manager builds the application manifest to let the end-user authorize which interactions the application can do. An application developer does not have to build a security manager. The permission can be installed by another application or another security manager.

In addition to the encrypted messaging (using AES CCM) between the peers, the Security 2.0 Permission Management module manages a database of access credentials and the Access Control Lists (ACLs).

Figure 2‑1 shows the system architecture of the Security 2.0 feature.



Figure 2‑1. Security system diagram

## Premises

Table 2‑1 lists the premises for the Security 2.0 features.

Table ‑. Security 2.0 premises

| Topic | Definition | Premises |
| --- | --- | --- |
| Identity | The application security principal | Each peer is identified by a keystore identity GUID and a cryptographic public key |
| Admin | An admin (or administrator) is a peer with administrator privilege for the application | * An admin has full access to any object and interface in the application * An admin becomes a certificate authority * An admin can be a public key or a security group |
| Claiming | Incorporate a factory-reset device with the Permission Management | * A factory-reset device has no list of certificate authorities for AllJoyn security. * A factory-reset device has no admin * Anyone can claim as an admin for a factory-reset device. |
| Policy | A policy is a list of rules governing the behavior of an application  A policy template is a list of rules defined by the application developer to guide the user for policy building.  A signed policy is a policy signed by an admin | * An admin can install, update, or remove a policy. * A newer policy can be installed by any authorized peer. Developers can define policy templates to help the user with policy building. * Security group specific policy specifies the permissions granted to members of the group. The security group authority becomes a certificate authority for that particular group. * A policy may exist at the producer or consumer side. Policy enforcement applies wherever it resides. * A policy is considered private. It is not exchanged with any peer. * An application has at most one policy. * An admin can query the existing policy installed in the application |
| Membership certificate | A membership certificate is the proof of a security group membership | * Membership certificates are exchanged between peers. The authorization data signed by this certificate are used for mutual authorization purposes. * An application trusts the membership certificate if the issuer or any subject in the issuer’s certificate chain matches any of the application’s certificate authorities. * A membership certificate holder can generate additional membership certificate for the given security group with the same or more restrictive permissions if the delegate flag is enabled. This type of membership certificate will not allow further delegation. * A membership certificate must have a security group ID. * An application can accept the installation of any number of membership certificates |
| Authorization data | The permission rules | * Authorization data are not present in the membership certificate * The membership authorization data is signed by the membership certificate issuer * Authorization data can be requested from the certificate holder. |
| Security group equivalence | The instruction on the policy on an application that allows applications from a different security manager to get the same access as a local security group | * An admin can add a security group equivalence instruction to the application. * Such instruction includes the public key of the other security manager. That public key becomes a certificate authority for authentication purpose. * Such instruction also includes the ID of the local security group. * All peer applications authenticated by this public key can get the same access to the ACLs defined by the specified security group. |
| Identity certificate | Certificate that signs the identity information. | * Certificate with a digest of the actual identity data. The identity data can be delivered out of band. * The Certificate has an alias field for that identity * An application trusts identity certificate issued by any of the application’s certificate authorities and security group equivalence authorities. |
| Security Manager |  | * Security Manager can push policy and certifcates to application |

## Typical operations

The following subsections describe the typical operations performed by a user.

### Claim a factory-reset application

Using the Security Manager any user can claim any factory-reset application. Claiming is a first-come, first-claim action. That user becomes the admin. The user can also install an admin security group. The procedure to make the application to become claimable again is manufacturer-specific. There will be an API call that allows the application to make itself claimable again.

#### Claim factory-reset application without out-of-band registration data



Figure 2‑2. Claim a factory-reset application without out-of-band registration data



Figure 2‑3: Claim a factory-reset application without using out-of-band registration data and install admin security group

The identity certificate will be used for authentication in the ECDHE\_ECDSA key exchange.

#### Claim factory-reset application using out-of-band registration data

An application manufacturer can provision a key to support the claiming process. The key is provided to the user out of band. An example is a QR code or a token delivered via email or text messaging. The user is prompted for the key when establishing a connection with the factory-reset application.



Figure 2‑4. Claiming a factory-reset application using out-of-band registration data

### Define a security group

Any user can define a security group (logical grouping of applications and users) using the Security Manager. When the user specifies a security group name (for display purpose), the Security Manager creates the security group ID (a GUID value).

### Example of building a policy

A user uses a Security Manager application to build a policy. The application queries the AllJoyn About feature data and the list of policy templates from the device. The Security Manager application can do further introspection of the device for the detailed information of secured interfaces and secured objects, and prompts the user to select the permissions to include in the policy.

A policy may contain a number of ACLs. Please refer to the section *Authorization data format* for more information.

### Install a policy

An admin can install a policy for the application.



Figure 2‑5. Install a policy

### Add an application to a security group

An admin signs a membership certificate with the given security group ID and installs it in the application. This act adds the application to the security group.



Figure 2‑6. Add an application to a security group

### Add a user to a security group

The security group authority uses the Security Manager to generate the membership certificate for the user for the given security group ID. The security group authority can remove some ACLs for this user.



Figure 2‑7. Add a user to a security group

### Delegating membership certificate

If a grantee receives a membership certificate with a delegate flag enabled, the grantee can issue a membership certificate to others with the same authorization or more restrictive authorization. Any peer validating a certificate chain verifies that no further delegation has been done, or the chain is considered invalid.



Figure ‑8. Reissue membership certificate

### Add a security group equivalence instruction to an application

An admin can update the permission policy in an application to add a security group equivalence instruction so the certificates issued by other certificate authorities (like friends) can be trusted. These friend’s applications would only have access to permissions assigned to the local security group specified in the instruction.



Figure ‑9. Add a security group equivalence instruction to an application

### Certificate revocation (not fully designed)

The application will validate the certificate using a revocation service provided by the Security Manager. The revocation service is a distributed service.

The Certificate Revocation Service is expected to provide a method call that takes in the certificate and return whether the given certificate is revoked.

The application looks in the “self” section of its installed policy for the peer that provides the Certificate Revocation Service. If the application can’t locate any of the Certificate Revocation Service, the certificate revocation check will be skipped.

If a membership certificate is revoked, all signed authorization data related to the membership certificate is no longer valid.

### Distribution of policy updates and membership certificates (not fully designed)

The Distribution Service is a service provided by a Security Manager. This service provides persistent storage and high availability to distribute updates to applications.

An admin uses the Security Manager to generate updated policy and membership certificates, encrypt the payload with a session key derived from a nonce value and the master secret for the <sender, recipient> pair. The package including the sender public key, recipient public key, nonce, and encrypted payload is sent to the Distribution Service to delivery to the recipient. The recipient uses the information in the package to locate the master secret to generate the corresponding session key to decrypt the payload. Once the decryption is successful, the recipient signs the hash of the package and provide the signature in the reply.



Figure 2‑8. Distribution of policy update and certificates

### Application Manifest (discussion-in-progress)

The main goal of a manifest is to inform the end-user which services an application will provide and consume. Manifest enforcement ensures the application cannot provide nor consume any unwarranted services. The trustworthy description of the interfaces shall be presented to the user in a human readable and localized fashion.

The manifest shall be enforced by the receiving peer, as a malicious application may not be trusted to enforce it locally.

#### Manifest Format

The format of the manifest is similar to the format of the authorization data. Please refer to the section *Policy Templates* below for more information.

#### Trusted Description

The manifest data provided by the application does not contain any description. The description would be provided via HTTPS by a Service Description Server:

1. Provided by the developer using the reserve domain name of the interface name
2. Provided by the AllSeen Alliance

The developer must at least provide the description for the interface. An interface member listed in the manifest should have a description. If there is no description for the member, the interface description will be used in its place.

#### Manifest enforcement

As manifest are incorporated in the membership policy, no additional enforcement mechanism is required. The remote peer will intersect the rules in its local policy with the rules defined in the membership policy to enforce the application manifest.

#### Generating Policy and Membership Based on Manifest

The following flow shows how the Security Manager uses the manifest data provided by the application to generate local policy and membership policies.

#### 

Figure 2‑9: Building Policy using manifest

## Access validation

### Validating a producer policy

This is a typical producer validation of the consumer permissions when the consumer makes a method call on a secure interface.



Figure 2‑10. Validating a producer policy

### Validating a consumer policy

This is a typical consumer policy validation when the consumer application calls a secure method call.



Figure 2‑11. Validating a consumer policy

### Exchanging membership certificates during session establishment

During the AllJoyn session establishment, the peers exchange all membership certificates. There is a potential information disclosure vulnerability. It is desired to have a more intelligent selection algorithm to provide membership certificates on demand and need-to-know basis. This algorithm needs to take into account of the latency of the certificate exchange during the method call invocation.



Figure 2‑12. Exchange membership certificates

### Anonymous session

In scenarios when there is no trust established between two peers such as when a guest comes into the user's home, the guest’s consumer application can still control certain applications if and only if there are ACLs specified for ANONYMOUS\_USER installed on these devices.

Note that ANY\_USER refers to authenticated peers while ANONUMOUS\_USER refers to unauthenticated peers.



Figure 2‑13. Anonymous access

### Validating an admin user



Figure 2‑14. Validating an admin user

### Emitting a session-based signal

Before emitting a session-based signal to existing connections, the producer verifies whether it is allowed to emit the given signal to any authorized party. Upon receipt of the signal, the consumer checks whether it has the authorization to accept the given signal. If the authorization is based on a security group, the consumer verifies the producer’s membership ACLs for proper authorization.



Figure 2‑15. Validating a session-based signal

## Authorization data format

### The format is binary and exchanged between peers using AllJoyn marshalling

The authorization data will be in binary format. The following guidelines are used for exchanging and persisting the authorization data:

1. The authorization data will use AllJoyn marshalling to exchange with other peers.
2. The AllJoyn marshalling will be used to generate buffer to be signed.
3. The AllJoyn marshalling will be used to serialize the data for persistence.
4. The parser will ignore any field that it does not support.

### Format Structure

The following diagram describes the format structure of the ACL data.



Figure 2‑16: Authorization Data Format Structure

#### Authorization data field definition

Root level

| Name | Data type | Required | Description |
| --- | --- | --- | --- |
| version | number | yes | The specification version number. The current spec version number is 1. |
| serialNumber | number | yes | The serial number of the policy. The serial number is used to detect of an update to an older policy. |
| ACLs | Array of ACLs | yes | List of access control lists. |

Access Control List

| Name | Data type | Required | Description |
| --- | --- | --- | --- |
| peers | array of objects | no | List of peers. There are multiple types of peers. A peer object has the following fields:   |  |  |  |  | | --- | --- | --- | --- | | **Name** | **Data**  **Type** | **Required** | **Description** | | type | number | yes | The peer type. The followings are the valid type of peers:   * ANONYMOUS (unauthenticated peer) * ANY\_USER (authenticated peer) * PUBLIC\_KEY * SECURITY\_GROUP | | keyInfo | structure of Authentication GUID and Public Key | no | The peer key info data. Depending on peer type, the keyInfo is:   * ANONYMOUS – not applicable * ANY\_USER – not applicable * PUBLIC\_KEY – the authentication GUID and public key of the peer * SECURITY\_GROUP – the authentication GUID and the public key of the security group authority | | sgID | GUID | No | Security group ID. This is application only the type SECURITY\_GROUP. | |
| rules | array of rules | no | List of allowed rules. The application is allowed to perform the actions specified in the given rules.  The default rule is to allow nothing. |

Rule Record

| Name | Data type | Required | List of values | Description |
| --- | --- | --- | --- | --- |
| obj | string | no |  | Object path of the secured object. A \* indicates a prefix match. When there is no \*, it is an exact match. |
| ifn | string | no |  | Interface name. A \* indicates a prefix match. When there is no \*, it is an exact match. |

Interface Member Record

| Name | Data type | Required | List of values | Description |
| --- | --- | --- | --- | --- |
| mbr | string | no |  | Member name. A \* indicates a prefix match. When there is no \*, it is an exact match. |
| type | number | no | * 1: method call * 2: signal * 3: property | Message type.  Default is method call. |
| action | byte | no |  | The action mask flag. The list of valid masks:   * 0x01: Denied * 0x02: Provide – allows sending signal, exposing method calls and producing properties * 0x04: Observe – allows receiving signals and getting properties * 0x08: Modify – set properties and make method calls |
| mutualAuth | boolean | no |  | Mutual authorization required. Both peers (local and remote) are required to be authorized. Specifying a no value means enabling a one-way authorization.  Default is yes. |

#### Enforcing the rules at message creation or receipt

The following table lists the required action mask base on the message.

Table -2: Action Mask Matrix

|  |  |  |
| --- | --- | --- |
| **Message Action** | **Local Policy**  **If there is no local policy, the default action is denied. Admin user has full access.** | **Remote peer’s membership ACL data.**  **Check when the authorization is security group specific.** |
| send GetProperty | Remote peer has PROVIDE permission for this property | Remote peer has PROVIDE permission for this property |
| receive GetProperty | Remote peer has OBSERVE permission for this property | Remote peer has OBSERVE permission for this property |
| send SetProperty | Remote peer has PROVIDE permission for this property | Remote peer has PROVIDE permission for this property |
| receive SetProperty | Remote peer has MODIFY permission for this property | Remote peer has MODIFY permission for this property |
| send method call | Remote peer has PROVIDE permission for this method call | Remote peer has PROVIDE permission for this method call |
| receive method call | Remote peer has MODIFY permission for this method call | Remote peer has MODIFY permission for this method call |
| send signal | Remote peer has OBSERVE permission for this signal | Remote peer has OBSERVE permission for this signal |
| receive signal | Remote peer has PROVIDE permission for this signal | Remote peer has PROVIDE permission for this signal |

#### Search Algorithm

Whenever an encrypted message is created or received, the authorization rules are searched using the message header data (object path, interface name, and member name) and the requested permission listed in Table 2-2: Action Mask Matrix.

#### Matching Algorithm within a Policy ACL

The following matching algorithm is used to find a match within a policy ACL. Once a match is found within the rules, the search stops.

* If the rule specifies both object path and interface name, the message must match both.
* If the rule specifies object path, the message must match the object path.
* If the rule specifies interface name, the message must match the interface name.
* If the rule specifies member name, the message must match the member name
* Verify whether the requested permission is allowed by the authorization mask at the member.
  + When a member name has an exact match and is explicitly denied access then the access is explicitly denied.
  + When a member name has an exact match and is authorized then the rule is a match
  + When a member name has a prefix match and is explicitly denied access then the rule is explicitly denied.
  + When a member name has a prefix match and is authorized then the rule is a match

#### Search Priorities for Policy ACLs

Policy ACLs are searched in this order. Once a match or an explicit deny is found, the search stops.

1. All public key policy ACLs
2. All security group policy ACLs are applied in undefined order. Per security-group-in-common, the materialized authorization rules are the intersection of the authorization rules between the consumer and producer.
3. The any-user policy ACLs.
4. The anonymous policy ACLs

### Policy Templates

An application developer can define policy templates to help the Security Manager to build consumer and producer policies. A policy template provides the following data in:

* Specification version number
* List of permission rules

## Certificates

The following subsections detail the supported certificates. The certificate format is X.509 v3. The certificate lifetime will be considered in order to avoid having to revoke the certificate. However, certain devices do not have access to a trusted real time clock. In such cases, the application on those devices will not be able to validate the certificate lifetime, thus relying on certificate revocation.

### Main Certificate Structure

All AllSeen X.509 certificates have the following ASN.1 structure. Currently only the ECDSA (prime256v1) certificates are supported.

Certificate ::= SEQUENCE {

tbsCertificate TBSCertificate,

signatureAlgorithm SEQUENCE { 1.2.840.10045.4.3.2 (ecdsa-with-sha256) },

signatureValue BIT STRING

}

TBSCertificate ::= SEQUENCE {

version v3(2),

serialNumber INTEGER,

signature SEQUENCE { 1.2.840.10045.4.3.2 (ecdsa-with-sha256) },

issuer SEQUENCE { 2.5.4.3 (commonName), UTF8 STRING },

validity Validity,

subject Name,

subjectPublicKeyInfo SEQUENCE { 1.2.840.10045.2.1 (id-ecPublicKey), 1.2.840.10045.3.1.7 (prime256v1), BIT STRING },

issuerUniqueID IMPLICIT UniqueIdentifier OPTIONAL,

subjectUniqueID IMPLICIT UniqueIdentifier OPTIONAL,

extensions EXPLICIT

}

#### Security 2.0 Custom OIDs

All Security 2.0 custom OIDs will start with 1.3.6.1.4.1.44924.1 where 1.3.6.1.4.1.44924 is the registered AllSeen Alliance Private Enterprise Number.

### Identity certificate

The identity certificate is used to associate application, user or device with an identity alias.

The alias is encoded in the SubjectAltName field in the extensions.

The extensions include the following fields:

* CertificateType: the type of certificate within the AllSeen ecosystem. An identity certificate has certificate type equal to 1.
* SubjectAltName: the alias for the identity.
* AssociatedDigest: the digest of the associated identity data. For example, an identity VCard.

Both the CertificateType and AssociatedDigest have custom OIDs under the Security 2.0 root.

SubjectName ::= SEQUENCE { 2.5.4.3 (commonName), UTF8 STRING },

Extensions ::= SEQUENCE {

BasicConstraints SEQUENCE { 2.5.29.19 (basicConstraints), BOOLEAN (FALSE) },

CertificateType SEQUENCE { 1.3.6.1.4.1.44924.1.1 (AllSeen Certificate Type), INTEGER (1) },

SubjectAltName SEQUENCE { 2.5.29.17 (subjectAltName), OCTET STRING },

AssociatedDigest SEQUENCE { 1.3.6.1.4.1.44924.1.2 (AllSeen Certificate Digest), 2.16.840.1.101.3.4.2.1 (sha-256), OCTET STRING }

}

### Membership certificate

The membership certificate is used to assert an application, user or device is part of a security group.

The security group identifier is encoded with a 16 network byte order octets in the Organization Unit Name within the Subject Distinguished Name field.

The extensions include the following fields:

* CertificateType: the type of certificate within the AllSeen ecosystem. A membership certificate has certificate type equal to 2.
* AssociatedDigest: the digest of the associated authorization data.

Both the CertificateType and AssociatedDigest have custom OIDs under the Security 2.0 root.

SubjectName ::= SEQUENCE { 2.5.4.11 (organizationalUnitName), UTF8 STRING, 2.5.4.3 (commonName), UTF8 STRING },

Extensions ::= SEQUENCE {

BasicConstraints SEQUENCE { 2.5.29.19 (basicConstraints), BOOLEAN default FALSE },

CertificateType SEQUENCE { 1.3.6.1.4.1.44924.1.1 (AllSeen Certificate Type), INTEGER (2) },

AssociatedDigest SEQUENCE { 1.3.6.1.4.1.44924.1.2 (AllSeen Certificate Digest), 2.16.840.1.101.3.4.2.1 (sha-256), OCTET STRING }

}

## Sample use cases

The solution listed here for the use cases is just a typical solution. It is not intended to be the only solution.

### Users and devices

Users: Dad, Mom, and son

| Room | Devices | Notes |
| --- | --- | --- |
| Living room | TV, Set-top box, tablet, Network-attached Storage (NAS) | * All devices owned by Dad * All devices are accessible for the whole family * Tablet is managed by Dad, but the whole family can use it |
| Son’s bedroom | TV | * Owned and managed by son * Devices are allowed to interact with living room devices but the parent al control feature is denied. |
| Master bedroom | TV, tablet | * TV used by Mom and Dad only * Tablet used by Dad only * Devices can interact with living room devices |

### Users set up by Dad



Figure 2‑17. Use case - users set up by Dad

### Living room set up by Dad



Figure 2‑18. Use case - living room set up by Dad

### Son's bedroom set up by son



Figure 2‑19. Use case - son's bedroom set up by son

### Master bedroom set up by Dad



Figure 2‑20. Use case - master bedroom set up by Dad

### Son can control different TVs in the house



Figure 2‑21. Use case – Son can control different TVs in the house

### Living room tablet controls TVs in the house



Figure 2‑22. Use case - Living room tablet controls TVs

# Enhancements to Existing Framework

## Crypto Agility Exchange

In order to provide the AllJoyn peers to express the desire to pick some particular cryptographic cypher suite to use in the key exchange and the encryption of the messages, new key exchange suite identifiers will be added to the framework to express the choice of cypher and MAC algorithms. The new identifiers may come from the list of TSL cipher suites specified in [Appendix A.5 of TLS RFC5246](http://tools.ietf.org/html/rfc5246#page-75) , [RFC6655](http://tools.ietf.org/html/rfc6655), and [RFC7251](http://tools.ietf.org/html/rfc7251).

The following table shows the list of existing key exchange suites:

|  |  |  |
| --- | --- | --- |
| **AllJoyn Key Exchange Suite** | **Crypto Parameters** | **Availability** |
| ALLJOYN\_ECDHE\_NULL | * Curve NIST P-256 (secp256r1) * AES\_128\_CCM\_8 * SHA256 | * Standard Client * Thin Client |
| ALLJOYN\_ECDHE\_PSK | * Curve NIST P-256 (secp256r1) * AES\_128\_CCM\_8 * SHA256 | * Standard Client * Thin Client |
| ALLJOYN\_ECDHE\_ECDSA | * Curve NIST P-256 (secp256r1) * AES\_128\_CCM\_8 * SHA256 * X.509 certificate | * Standard Client * Thin Client |
| ALLJOYN\_RSA\_KEYX | * AES\_128\_CCM\_8 * SHA256 * X.509 certificate | * Standard Client |
| ALLJOYN\_PIN\_KEYX | * AES\_128\_CCM\_8 | * Standard Client * Thin Client version 14.02 or older |
| ALLJOYN\_SRP\_KEYX | * AES\_128\_CCM\_8 | * Standard Client |
| ALLJOYN\_SRP\_LOGON | * AES\_128\_CCM\_8 | * Standard Client |

The following table shows the potential list of TLS cipher suites to be supported. Other suites will be added as codes are available.

|  |  |  |  |
| --- | --- | --- | --- |
| **TLS cipher suite** | **Additional Crypto Parameters** | **Availability** | **RFC** |
| TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CCM\_8 | * Curve NIST P-256 (secp256r1) * SHA256 * X.509 certificate | * Standard Client * Thin Client | [7251](http://tools.ietf.org/html/rfc7251) |
| TLS\_RSA\_WITH\_AES\_128\_CCM\_8 | * SHA256 * X.509 certificate | * Standard Client | [6655](http://tools.ietf.org/html/rfc6655) |

## Permission NotifyConfig Announcement

The Permission module provides a session-less signal to allow the Security Manager discovering the applications to claim or to distribute updated policy or certificates. The signal provides the following information:

1. A number field named **claimable** to show the claim state of the application. The possible values of this field are:
   * + 0 -- not claimable
     + 1 – claimable
     + 2 - claimed
2. The authentication GUID and public key
3. The permission policy serial number
4. The list of acceptable key exchange suites for claiming

This signal is emitted when

1. The bus attachment is enabled with peer security using ECDHE key exchanges
2. The application is claimed or do a factory reset
3. The application has a permission policy installed
4. The application has its permission policy removed

# Future Considerations

## Broadcast signals and multipoint sessions

All security enhancements for broadcast signals and multipoint sessions will be considered in future releases of Security 2.0.