AllJoyn™ Security 2.0 Feature

High-Level Design Document

Rev 1 Update 1

August 27, 2014

Contents

Figures

Tables

1 Introduction

1.1 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.

1.2 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

1.3 Acronyms and terms

Acronym/term	Description
About data	Data from the About feature. For more information, refer to the About Feature Interface Spec.
ACL	Access Control List
AES CCM	The Advanced Encryption Standard 128-bit block cypher using Counter with CBC-MAC mode. Refer to RFC 3610 for more information.
Provider	An AllJoyn application advertises its interfaces so other AllJoyn application may access/control it.
Consumer	An AllJoyn application which is able to control or uses services provided by another AllJoyn application.
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

Acronym/term	Description
Grantee	The application or user who is the subject of a certificate.
GUID	Globally Unique Identifier
Guild	A logical grouping of devices, applications, and users. It is identified by a guild ID which is a GUID. An application can be installed with a policy to expose services to members of the guild. An application or user holding a membership certificate is in fact a member of the guild. Any member of the guild can access the services exposed to the guild by the applications with policies defined for that guild.
Guild Authority	A guild authority is the user or application that defines the guild policy and grant membership certificates to other. The guild authority is the certificate authority for that guild.
Holder	The application or user possessing a certificate.
Issuer	The application or user signing 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	A set of AllJoyn interfaces to manage cryptographic keys, generate and distribute certificates.
Security Appliance	A security appliance is a type of Security Manager that is always present.
IoE	Internet of Everything
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

2 System Design

2.1 Overview

The goal of the Security 2.0 feature is to allow an application to validate access to secure 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 authorization before any message action can be taken.

The Security Manager is optional service that helps the user with key management and permission rules building. Using policy templates defined by application developer, the Security Manager builds the application manifest to let the end-user authorize which interactions the application can do. The security Manger is optional because the permissions can be installed directly into the application.

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.

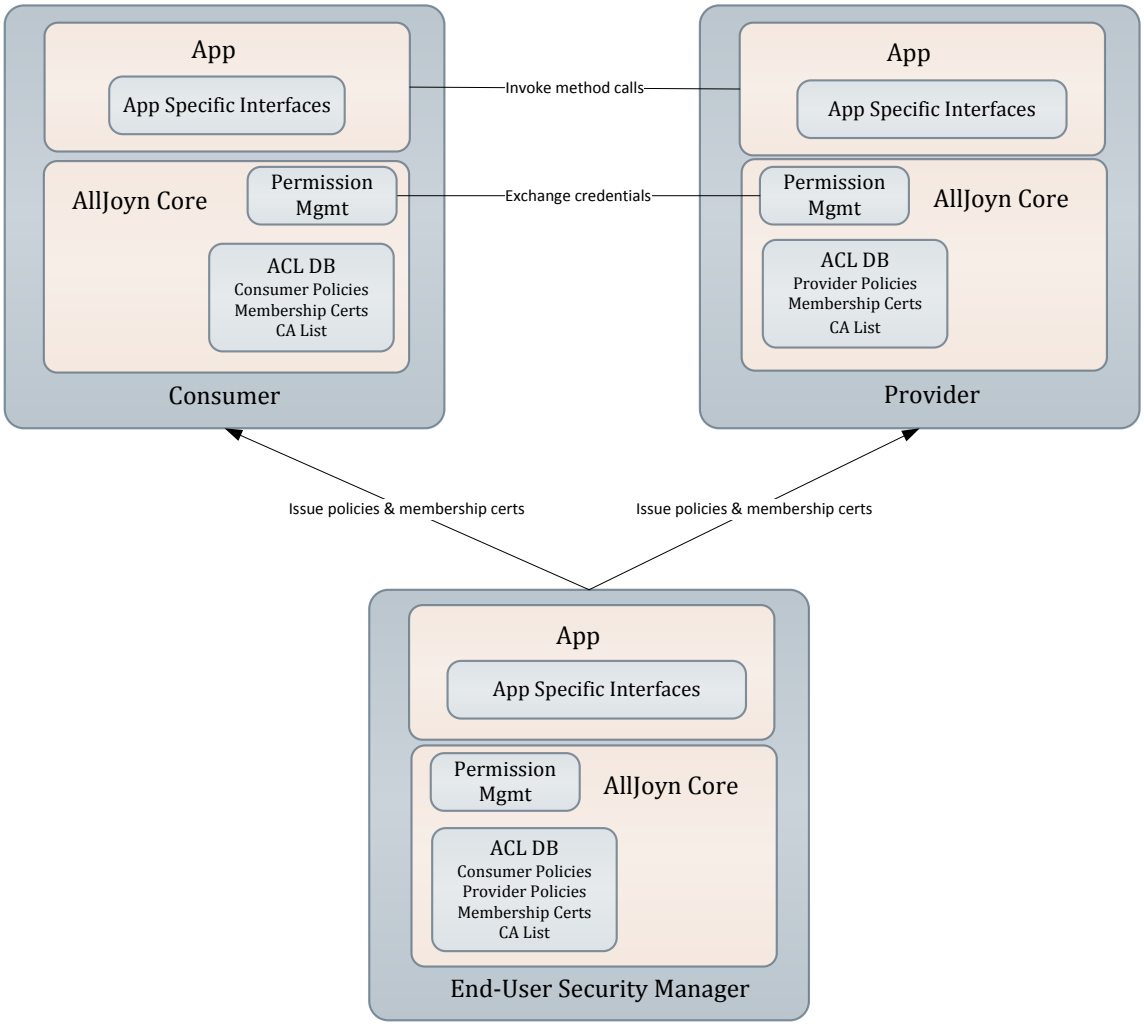

Figure 2-1. Security system diagram

2.2 Premises

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

Table 2-1. Security 2.0 premises

Topic	Definition	Premises
Identity	The application identification	All peers are identified by 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 add/remove another admin

Topic	Definition	Premises
Claiming	Incorporate a factory-reset device with the Permission Management	■ A factory-reset device has no list of certificate authorities. ■ 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 signed policy can be installed by any peer . ■ Developers can define policy templates to help the user with policy building. ■ Guild-specific policy specifies the permissions granted to members of the guild. The guild authority becomes a certificate authority for that particular guild. ■ A policy may exist at the provider or consumer side. Policy enforcement applies wherever it resides. ■ A policy is considered private. It is not exchanged with any peer. ■ An application may zero or more policies. ■ An admin can query the existing policy installed in the application
Membership certificate	A membership certificate is the proof of a guild 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 certificate authorities. ■ A membership certificate holder can generate additional membership certificate for the given guild 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 guild ID. ■ A device or application can accept any number of membership certificates
User equivalent certificate	A user equivalence certificate allows the holder to act like the issuer	The holder has the same access rights as the issuer
Authorization data	The permission rules	■ Authorization data are not present in the membership or the policy certificate ■ The certificate holds the digest of the authorization data. ■ Authorization data can be requested from the certificate holder.
Guild equivalence certificate	Certificate maps other guilds to a specific guild	■ An admin can add a guild equivalence certificate to the application. This mechanism allows other guilds to map to a specific guild. ■ The subject in the certificate is the equivalence guild authority's public key. A membership certificate generated from that guild authority or its delegates will have access to the specific guild defined in the guild equivalence cert.

Topic	Definition	Premises
Identity certificate	Certificate that signs the identity information and optional vCard data.	■ Certificate with a digest of the actual identity data. ■ The Certificate has an alias field for that identity in addition to the external Identification data ■ A peer can request for the other peer's identity certificate and identity data. ■ An application trusts identity certificate issued by any of the application's certificate authorities and guild equivalence authorities. ■ An application may have one or more identity certificates.
Security Manager		■ Security Manager can push signed policy to subject application ■ Security Manager provides interface for application to retrieve policy updates

2.3 Typical operations

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

2.3.1 Claim a factory-reset device

A user can claim any factory reset device during the claiming interval. Claiming is first-come, first-claim action. That user becomes the admin. If there is no claimant during the claiming interval, the device becomes unclaimable. The procedure to make the device to become claimable again is manufacturer's specific.

2.3.1.1 Claim factory-reset device without out-of-band registration data

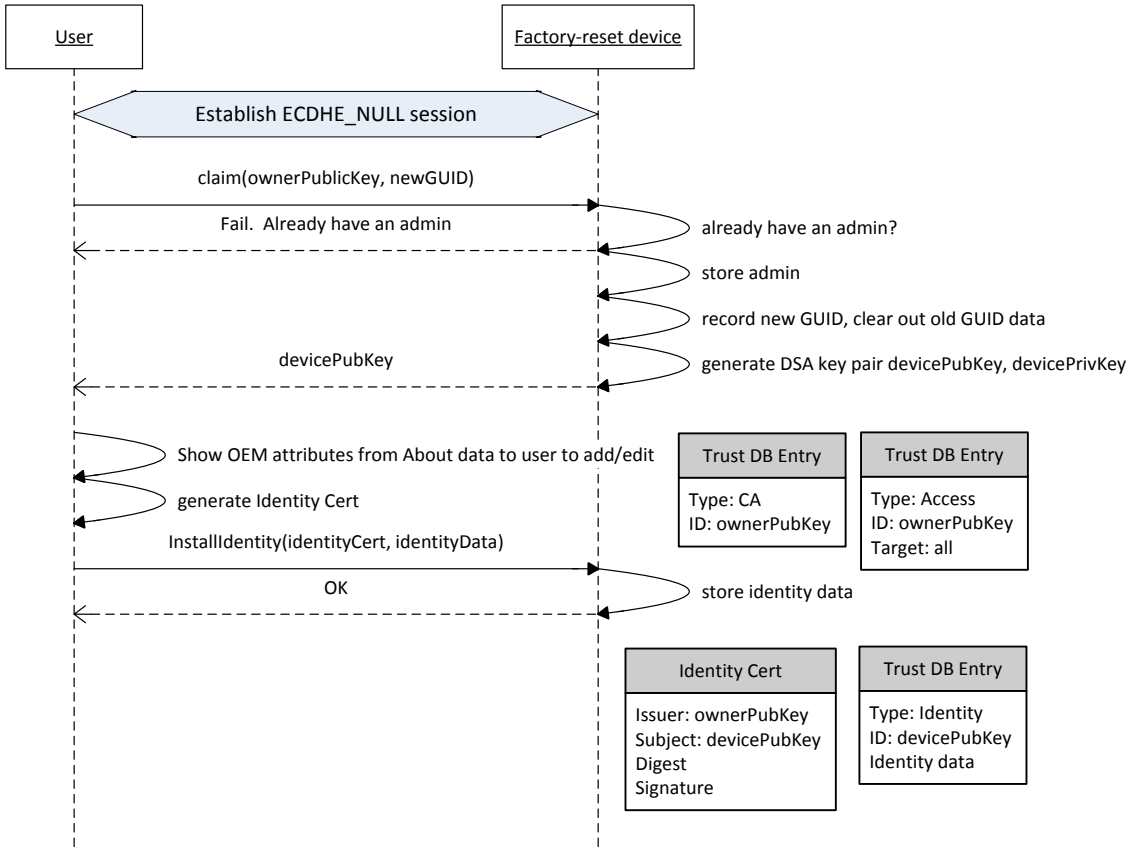

Figure 2-2. Claim a factory-reset device without out-of-band registration data

2.3.1.2 Claim factory-reset device using out-of-band registration data

A device 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 establish connection with the device.

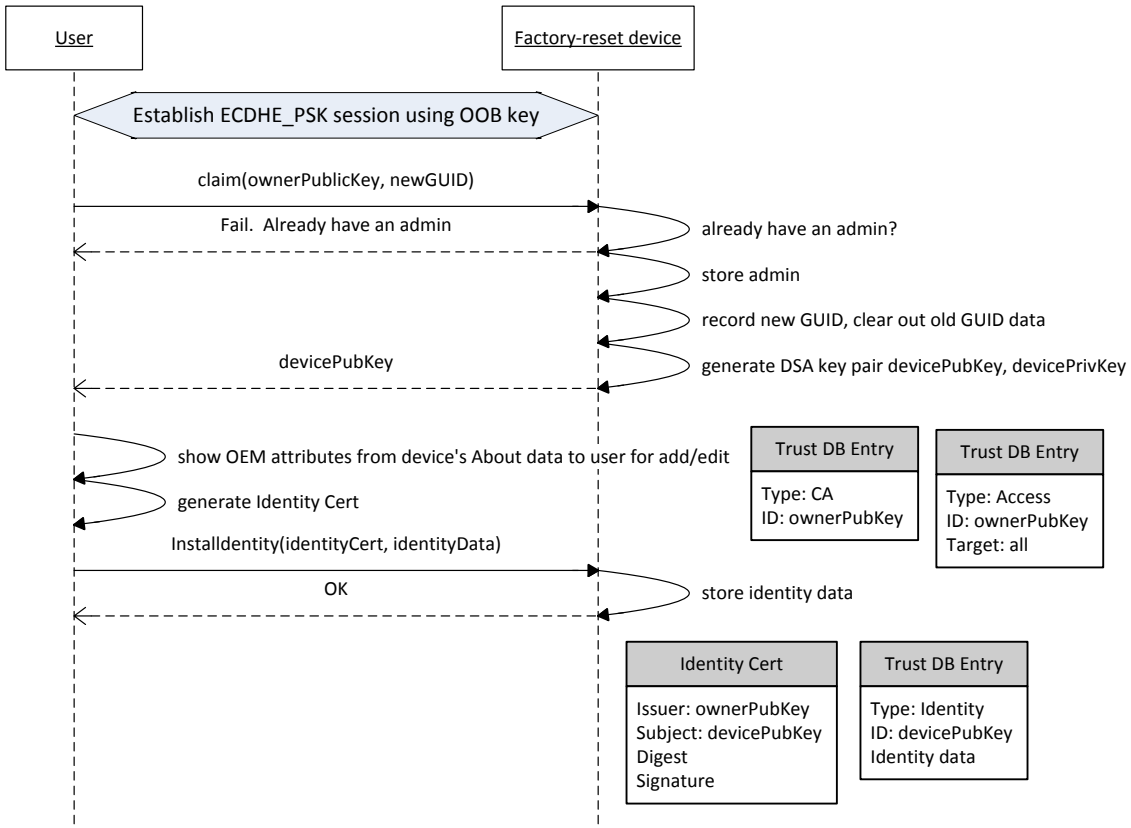

Figure 2-3. Claiming a factory-reset device using out-of-band registration data

2.3.2 Define a guild

A user can define a guild (logical grouping of devices and users) using the Security Manager. When the user specifies a guild name, the Security Manager creates the guild ID (typically a GUID value).

2.3.3 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.

2.3.4 Install an ANY-USER policy

An admin can install an ANY-USER policy for the application. This policy specifies all the authorizations for any user.

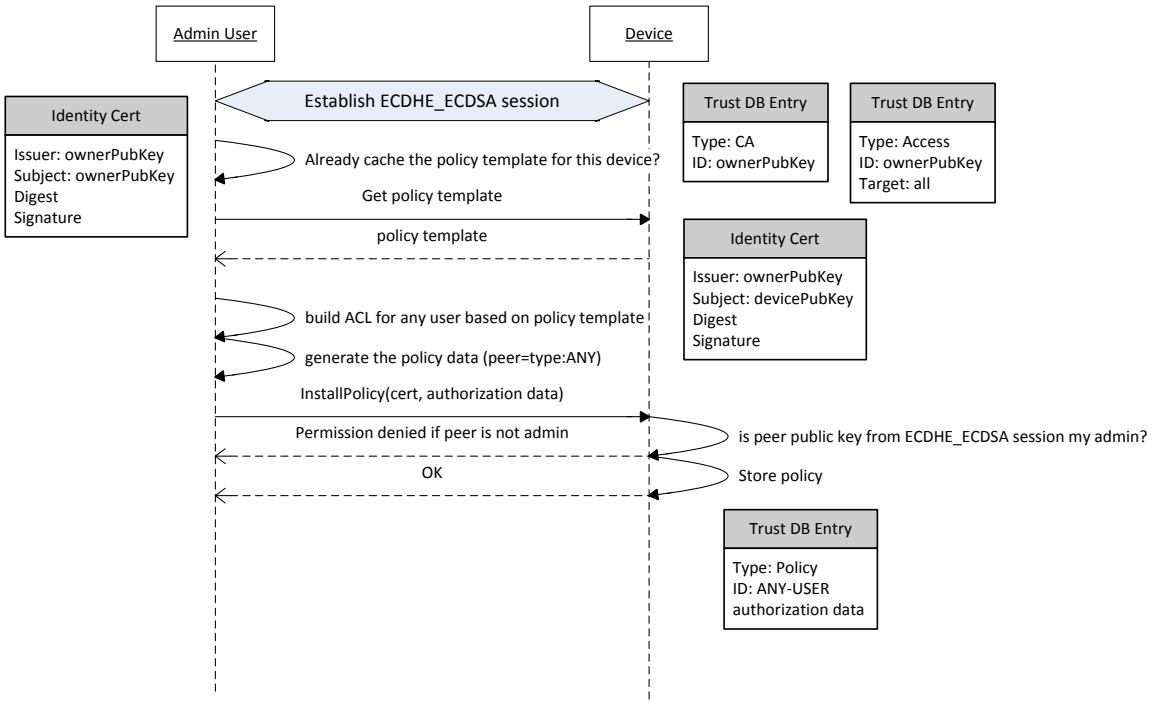

Figure 2-4. Install an ANY-USER policy

2.3.5 Install a guild-specific policy

An admin can install a guild-specific policy on the application. This policy specifies how the given application behaves based on its role as a provider and/or a consumer. See Authorization data format for more details on the format of the authorization data. Typically, a provider has policies installed, but occasionally a consumer may also have a policy. For a provider, the policy specifies all the authorizations to any consumer presenting a guild membership certificate. For a consumer, the policy specifies whether it has the privilege to send a command to the provider or to receive a signal from the provider.

Installing a guild-specific policy does not allow the holding application to access other applications in the guild.

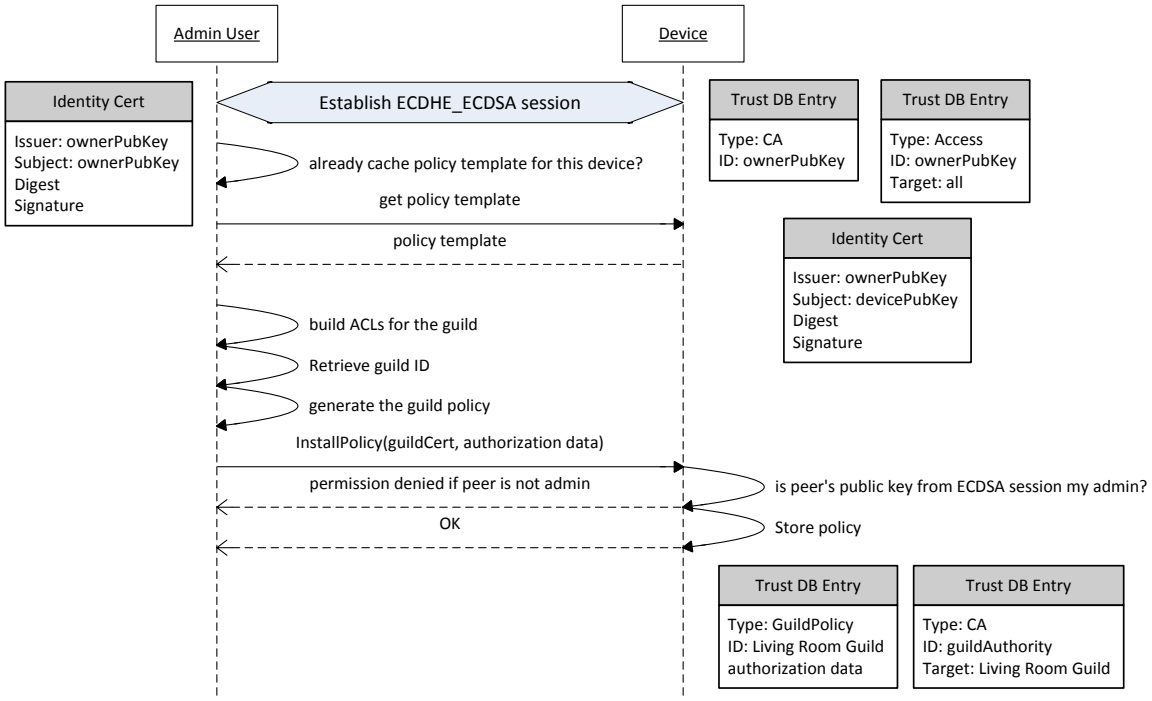

Figure 2-5. Install a guild-specific policy

2.3.6 Add an application to a guild

An admin signs a membership certificate with the given guild ID and installs it in the application. This act adds the application to the guild. In order for a provider to emit signals to other members of the guild, the provider must have a membership certificate with proper authorization to do so.

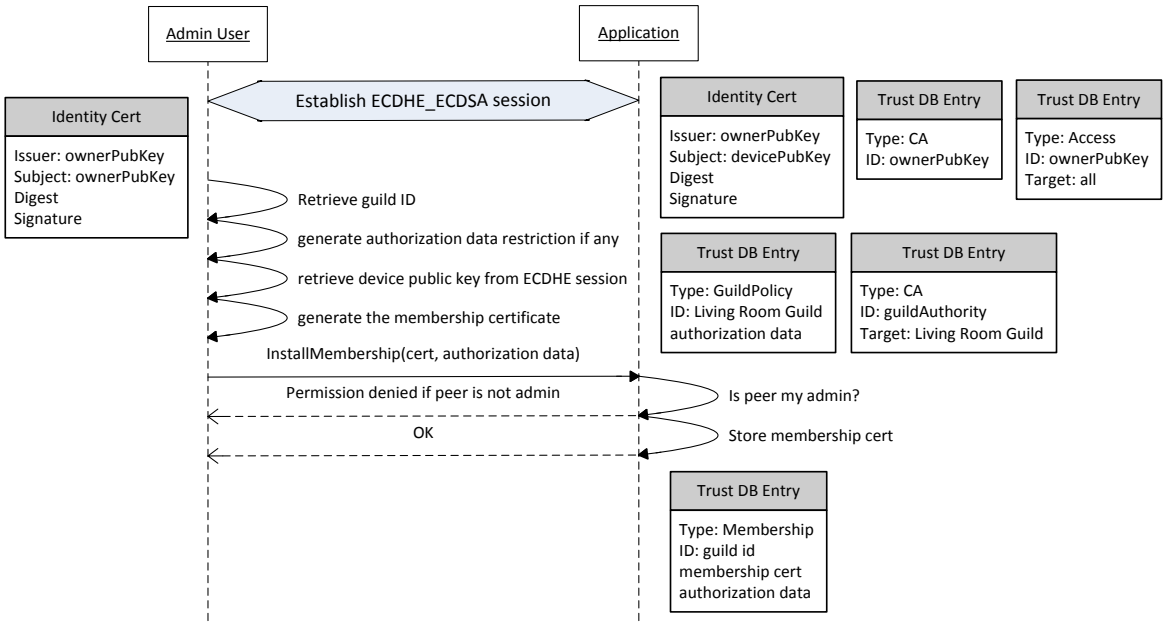

Figure 2-6. Add an application to a guild

2.3.7 Add a user to a guild

The guild authority uses the Security Manager to generate the membership certificate for the user for the given guild ID. The guild authority can restrict the permissions for this user.

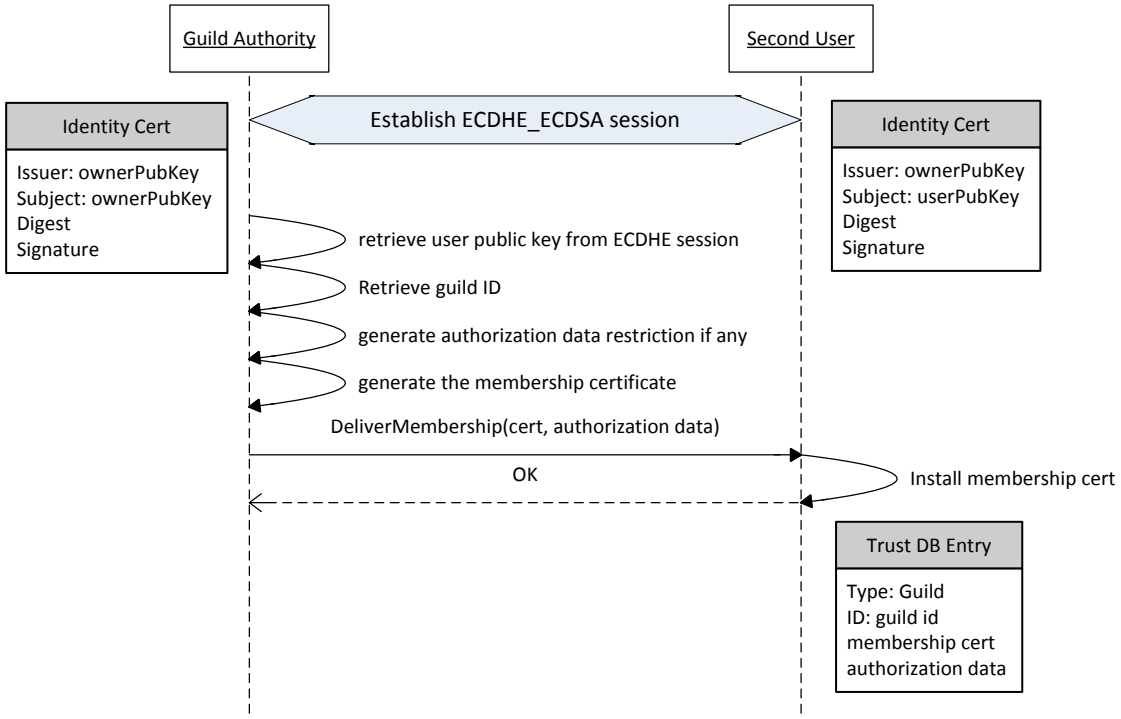

Figure 2-7. Add a user to a guild

2.3.8 Delegating membership certificate

If a grantee receives a membership certificate with a delegate flag enabled, the grantee can issue the same membership certificate to others with the same authorization or more restrictive authorization. The peer verifies that no further delegation is allowed.

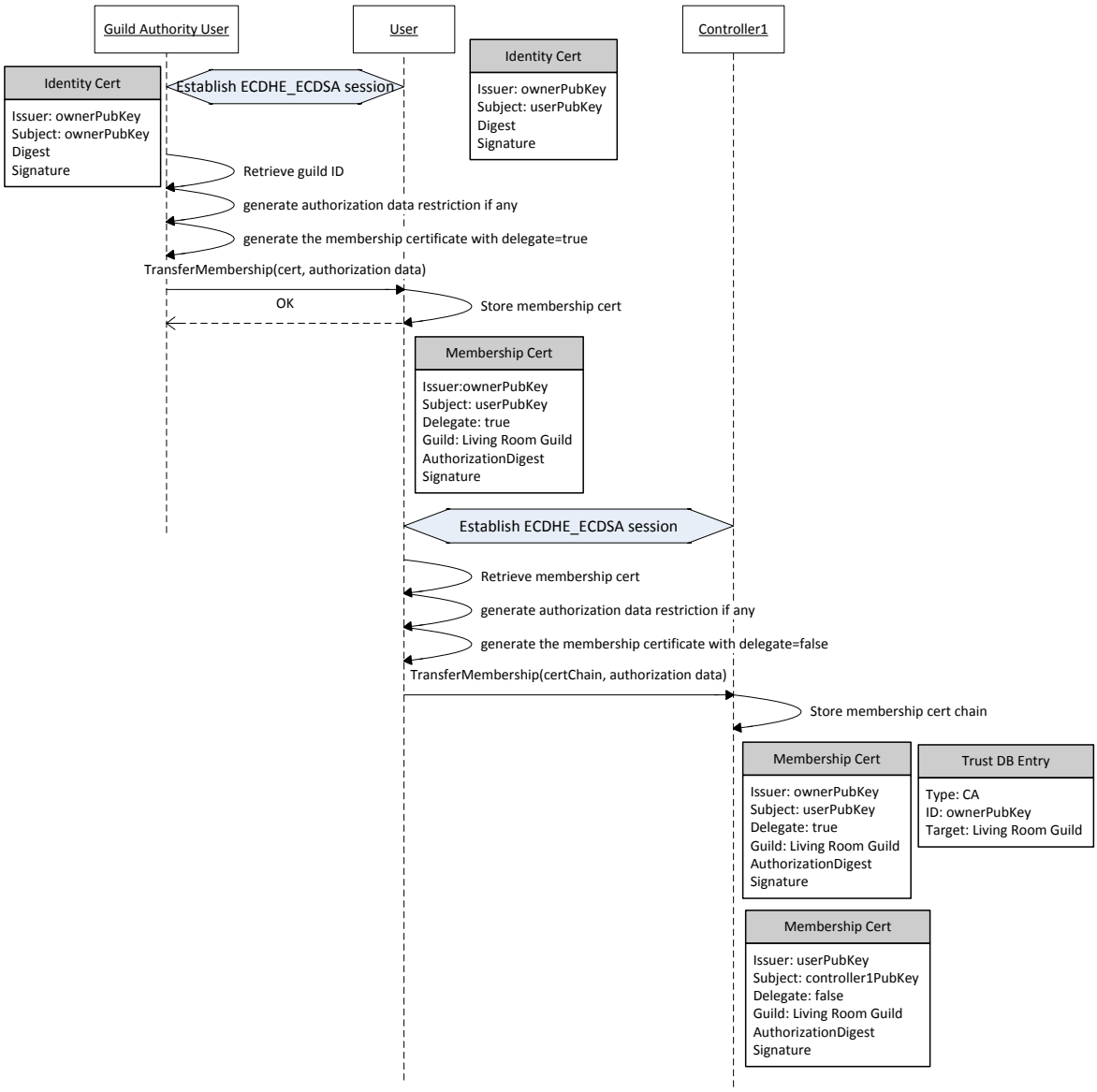

Figure 2-8. Reissue membership certificate

2.3.9 Add a guild equivalence certificate to an application

An admin can add a guild equivalence certificate to the application so the membership certificates issued by other certificate authorities (like friends) can be trusted. These certificate holders would only have access to permissions assigned to that specific guild.

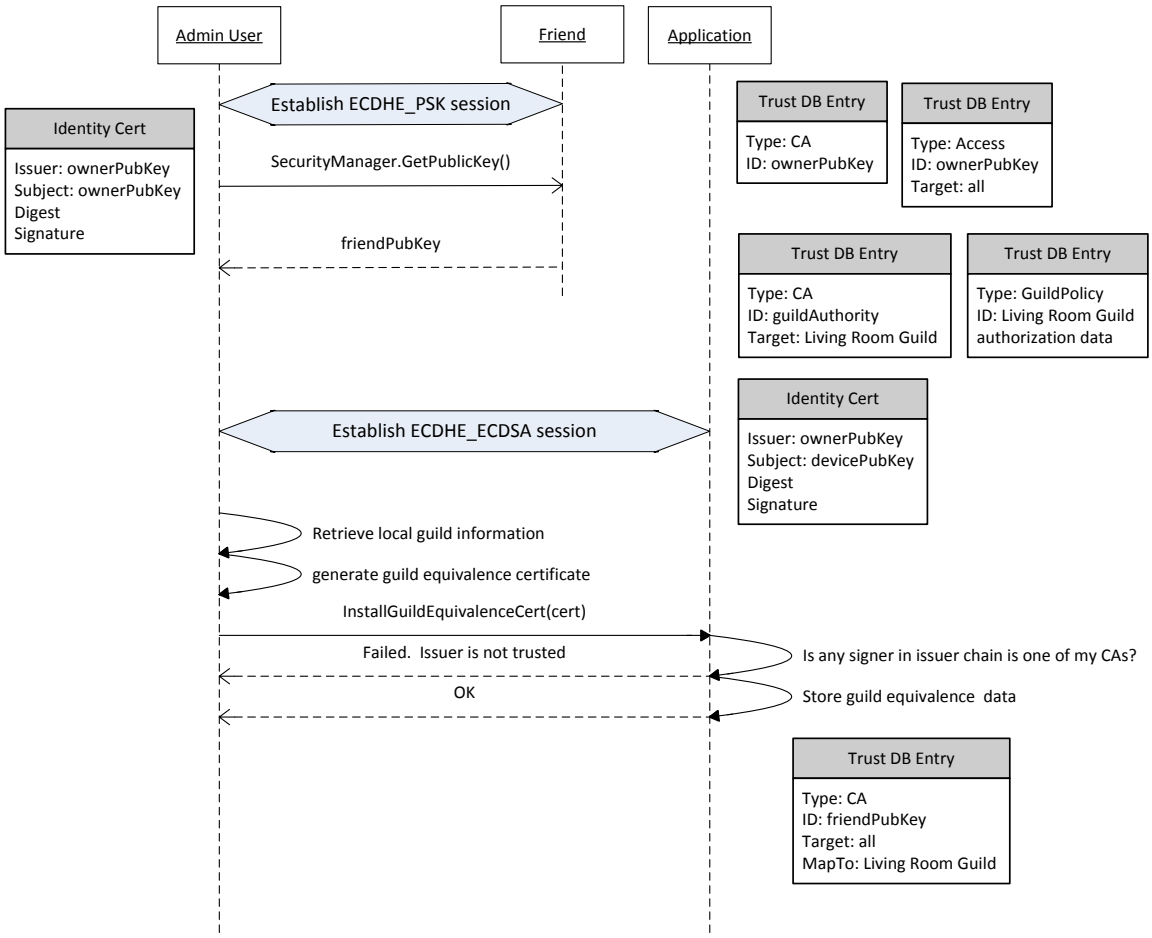

Figure 2-9. Add a guild equivalence certificate to an application

2.3.10 Certificate revocation

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

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

The application checks its installed policy for the Security Manager guild. If there is no such guild or the application can't locate any of the Security Manager, 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. The relationship is defined by the matching issuer public key, subject public key, and guild ID.

2.3.11 Distribution of policy updates and membership certificates

An admin uses the Security Manager to generate updated policy and membership certificates and deliver them to the Distribution Service for distribution to the applications he/she owns. These policy updates and certificates are signed by an admin, and the subject is the specific application. As the result, the application will trust the certificate and the signed policy updates.

The Distribution Service is a service provided by the Security Appliance or the Security Manager. This service provides persistent storage and high availability to allow for push and pull strategy to distribute updates to applications.

The Distribution Service broadcasts that updates are available so the applications can connect to it in order to retrieve the updates. The Distribution Service attempts to install the updated certificates if any application does not retrieve its updates after a certain time period.

A typical notification message from the Distribution Service contains the issuer's public key, peerID or guild ID, and version information.

2.3.12 Application Manifest (Work-in-progress)

This is place holder for the description of the application manifest and process.

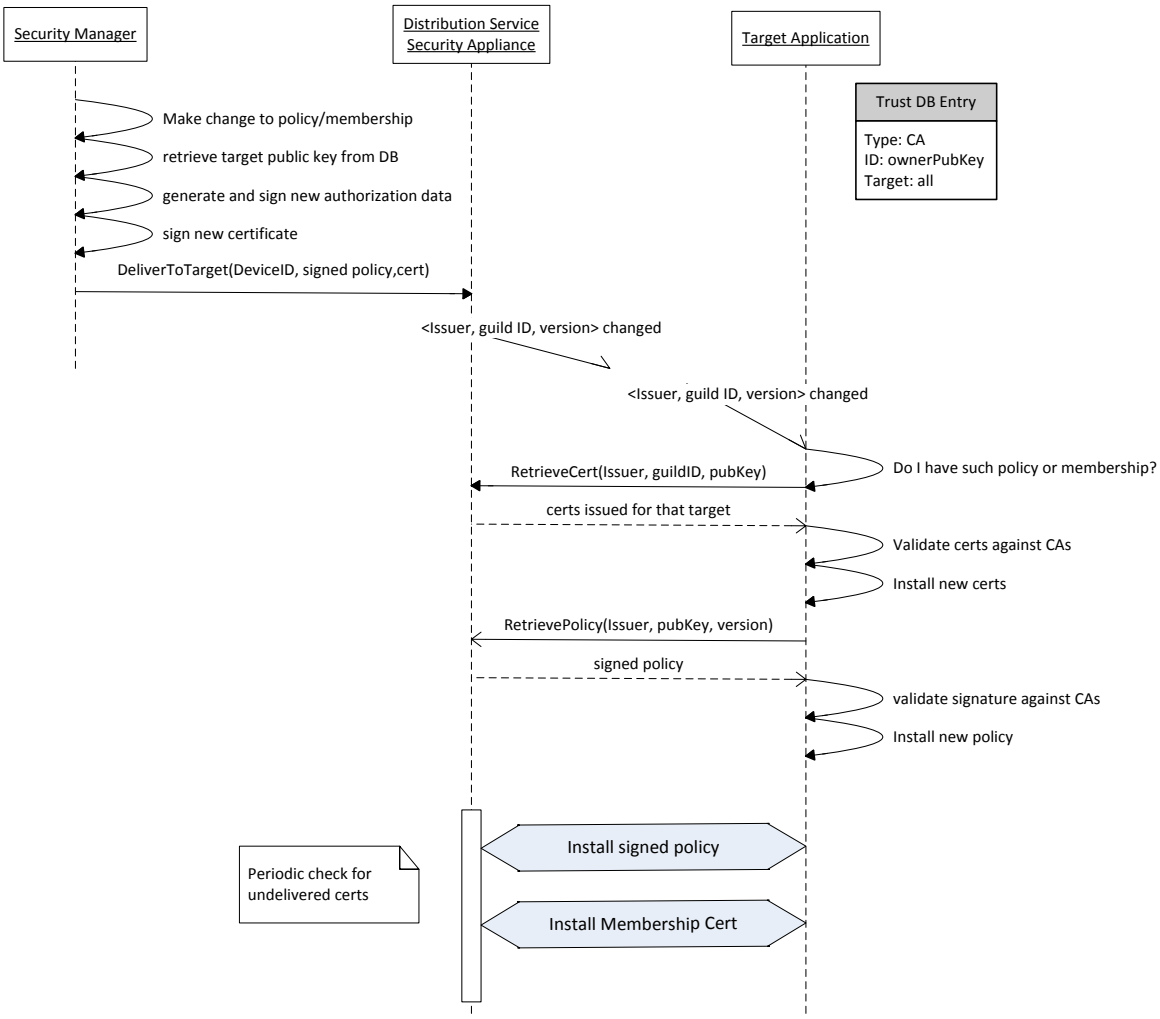

Figure 2-8. Distribution of policy updates and certificate

2.4 Access validation

2.4.1 Validation flow

A typical provider validation of the consumer permissions when a secure interface is requested.

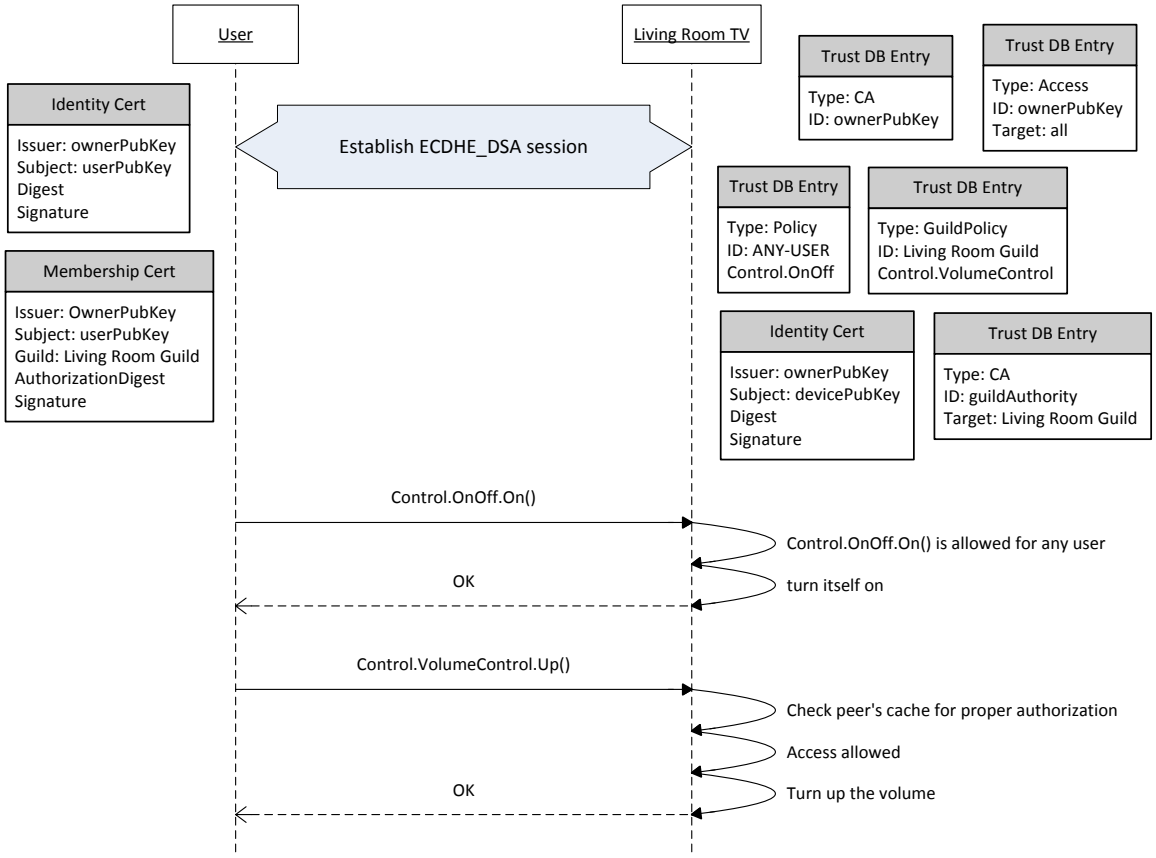

Figure 2-9. Validation Flow

2.4.2 Validating a consumer policy

A typical consumer policy validation when a secure method call is called by the consumer's app.

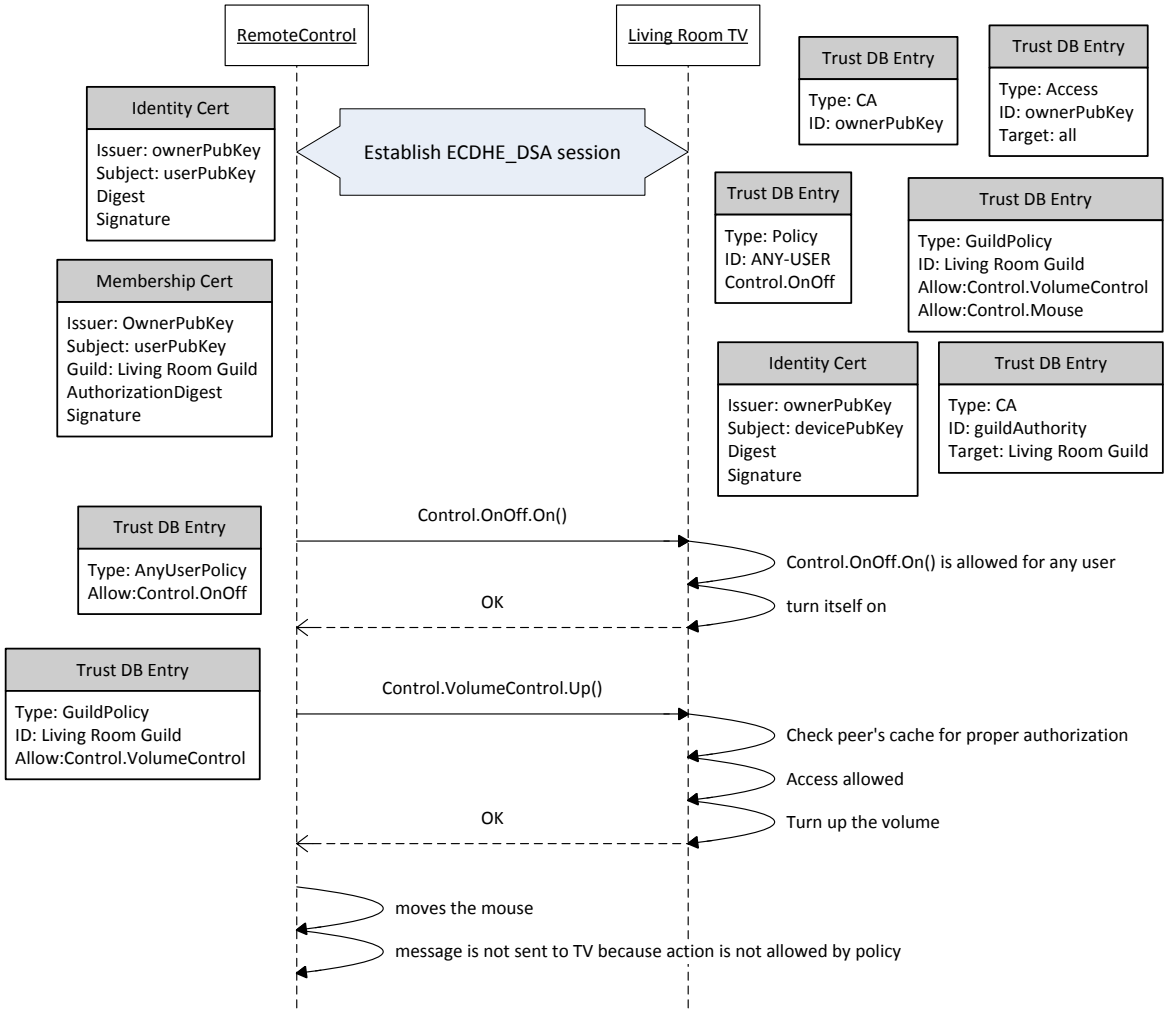

Figure 2-10. Validating a consumer policy

2.4.3 Exchanging a trust profile during session establishment

During the AllJoyn session establishment, the peers exchange the trust profile to determine what they have in common.

- A consumer's trust profile typically holds the list of membership guilds (guild ID and membership cert signer chain), at least one certificate (typically it's the identity certificate) with the issuer's certificate chain attached.
- A provider's trust profile typically holds the list of policy guilds (guild ID and guild-authority), membership guilds (guild ID and membership cert signer chain), at least one certificate (typically it's the identity certificate) with the issuer's certificate chain attached.

This process is initiated by the session joiner side (typically it's the consumer side). Based on the trust profile data received, the consumer determines which membership certificates to send to the provider. The consumer also determines which membership certificates it requests from the provider. The access data are cached on both sides to enforce the authorization rules as the message comes in.

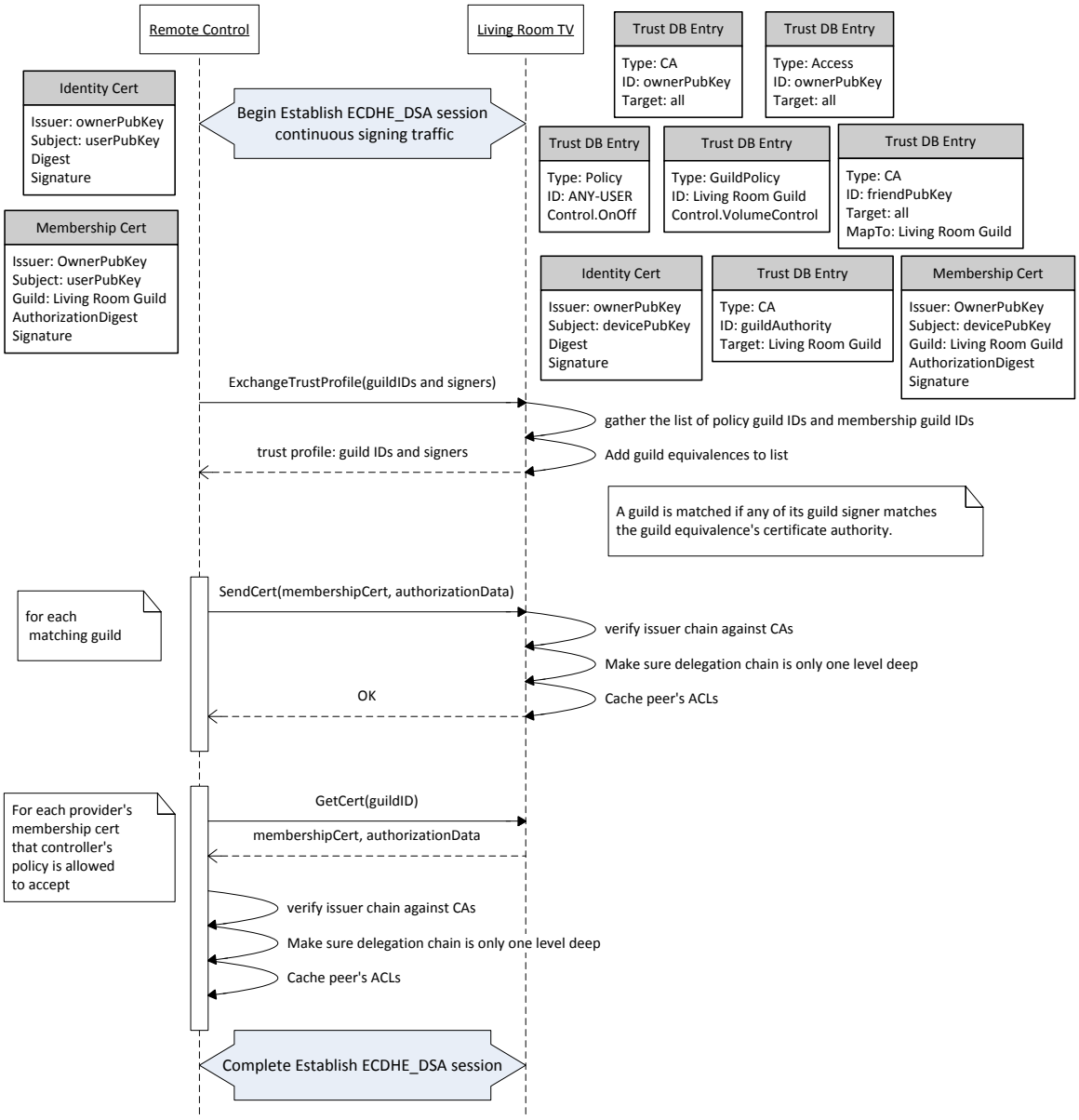

Figure 2-11. Exchange a trust profile

2.4.4 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 devices if and only if there is an ANY_USER policy installed on these devices. In such a scenario, the consumer application can ask the Permission Management module to switch to an ECDHE_NULL session for a short period of time.

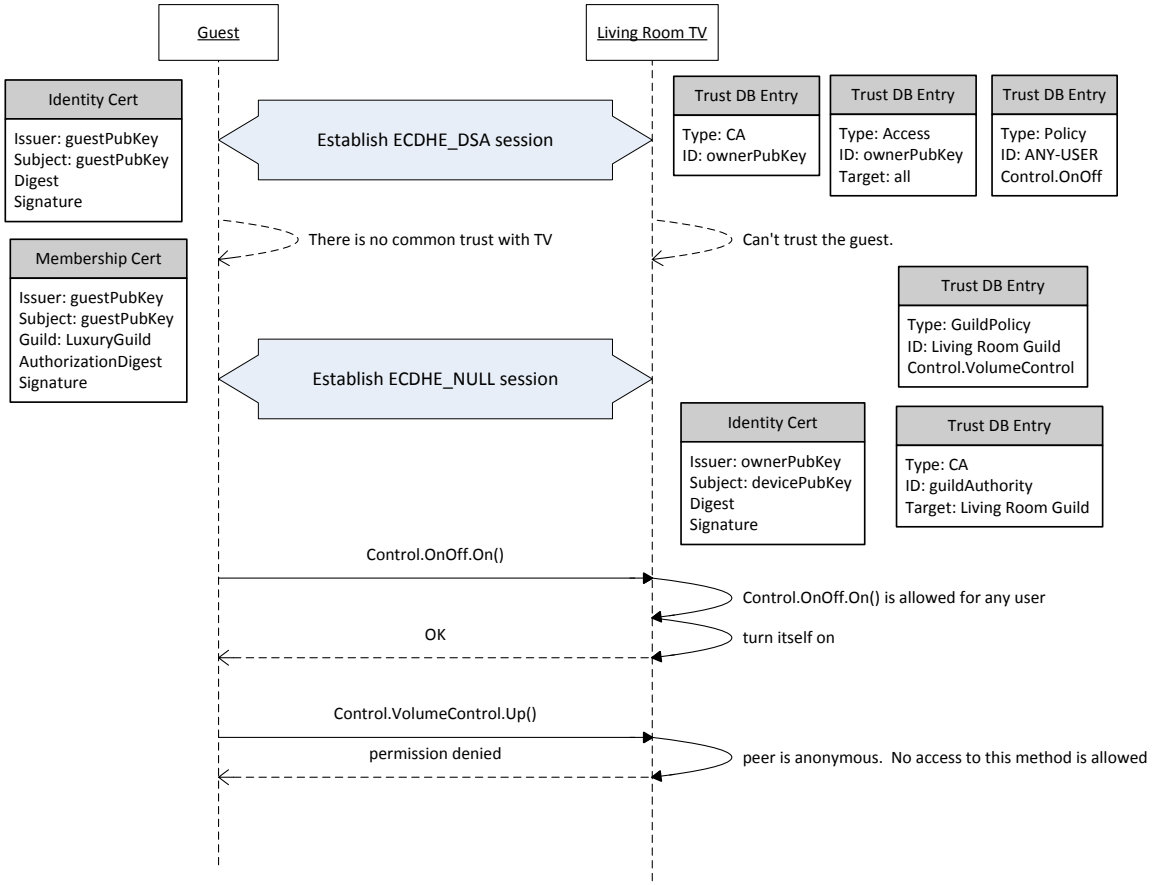

Figure 2-12. Anonymous access

2.4.5 Validating an admin user

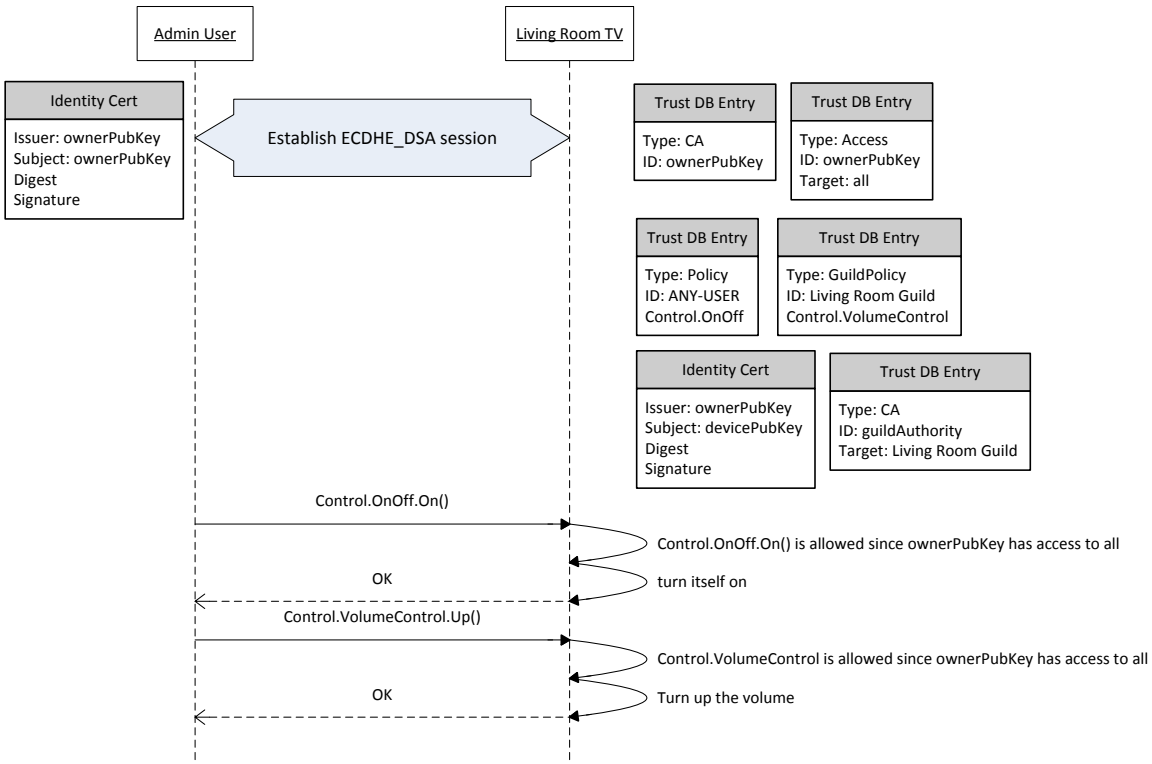

Figure 2-13. Validating an admin user

2.4.6 Emitting a session-based signal

Before emitting a session-based signal to existing connections, the provider verifies whether it is allowed to emit the given signal to the guild members. The provider also verifies that the recipient is authorized to receive the signal. Upon receipt of the signal, the consumer checks whether it has a policy to allow it to accept the given signal. The consumer also checks whether the provider is authorized to emit the given signal.

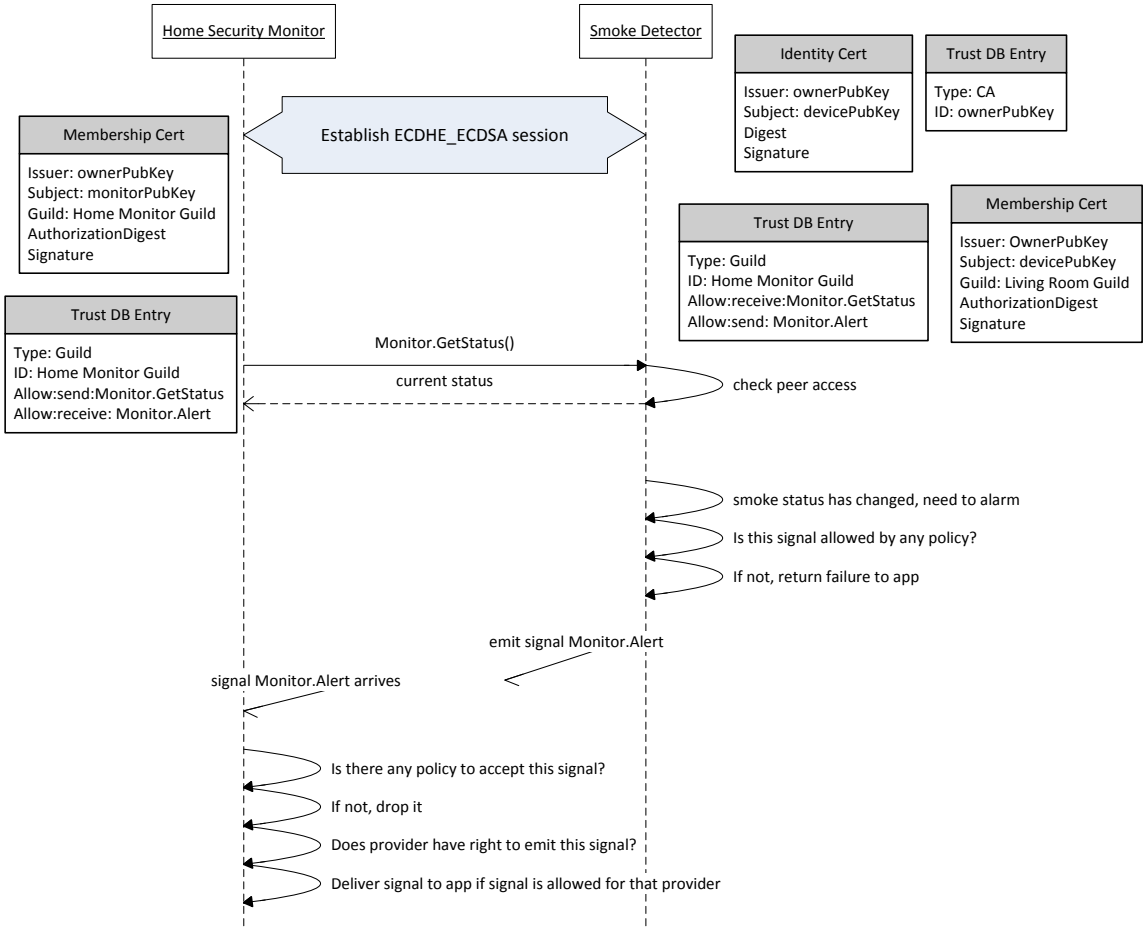

Figure 2-14. Validating a session-based signal

2.5 Authorization data format

2.5.1 The format syntax is not yet agreed

The format syntax of the authorization data is not yet agreed. As the result, in this section, the examples are listed in some tabular format as illustration.

Three proposed formats are:

1. Use AllJoyn message encoding to encode the authorization data. This will eliminate the needs of additional parsing codes in the application. The application uses its own methodology to persist the data for exchange with other peers.
2. Expressed the authorization data in JSON

3. Expressed the authorization data in XACML (eXtensible Access Control Markup Language). XACML is very much verbose than JSON.

2.5.2 Format Structure

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

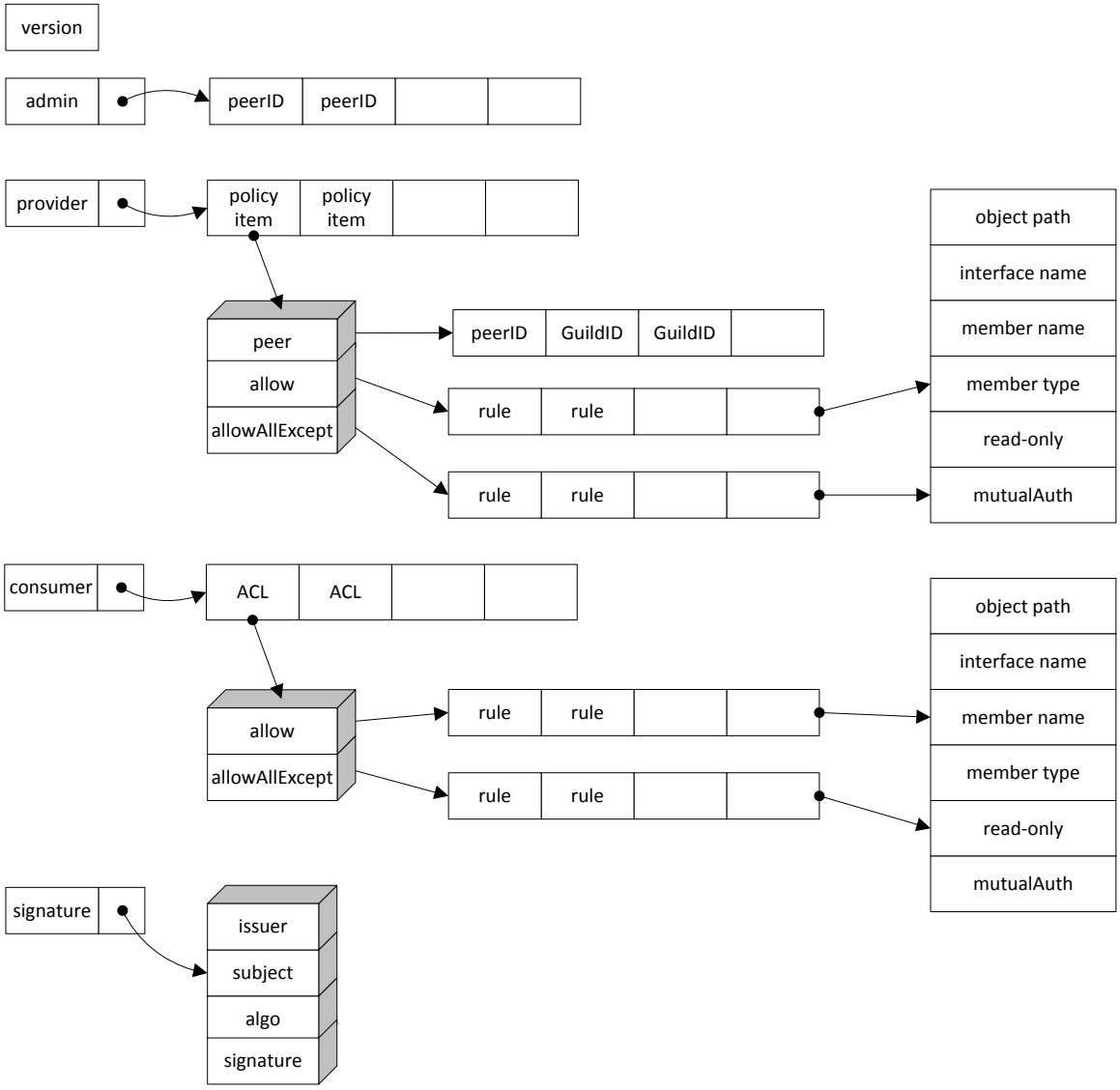

Figure 2-15: Authorization Data Format Structure

2.5.2.1 Authorization data field definition

Root level

Name	Data type	Required	Description
version	number	yes	The version number of the policy
admin	Array of peer objects	no	The list of peers who have the admin privilege for the application. An admin peer becomes a certificate authority for the application.
provider	Array of Policy Item objects	no	List of provider policy items. A provider policy specifies the features the application can provide to others.
consumer	Array of ACL objects	no	List of consumer ACLs. A consumer ACL specifies the features the application can invoke on others.
signature	Signature object	no	The signature data if the policy needs to be signed by an admin

Policy Item/ACL

<table>
<tr><th>Name</th><th>Data type</th><th>Required</th><th>Description</th></tr>
<tr><td>peer</td><td>array of objects</td><td>no</td><td>List of peers. There are multiple types of peers. A peer object has the following fields:
<table>
<tr><th>Name</th><th>Data Type</th><th>Required</th><th>Description</th></tr>
<tr><td>type</td><td>string</td><td>yes</td><td>The peer type. The followings are the valid type of peers:
ANY
PSK
DSA
GUILD</td></tr>
<tr><td>ID</td><td>string</td><td>no</td><td>The peer ID. Depending on peer type, the ID is:
ANY – not applicable
PSK – the PSK name
DSA – the public key
GUILD – the GUID of the guild</td></tr>
<tr><td>authority</td><td>string</td><td>no</td><td>The guild authority. This field is applicable for a guild.
The guild authority becomes a certificate authority for the given guild.</td></tr>
<tr><td>psk</td><td>string</td><td>no</td><td>The hex encoded shared secret. It is applicable for peer type PSK.</td></tr>
</table></td></tr>
<tr><td>allow</td><td>array of rules</td><td>no</td><td>List of allowed rules. The application is allowed to perform the actions specified in the given rules.

The default rule is to allow nothing.</td></tr>
<tr><td>allowAllExcept</td><td>array of rules</td><td>no</td><td>A short cut to say allow all except these rules. This field is ignored if the allow field is present and different than “*”</td></tr>
</table>

Rule used in Allow or AllowAllExcept record

Name	Data type	Required	List of values	Description
obj	string	no		Object path of the secured object. A * indicates a prefix match. **If the object path is specified, the remaining fields are ignored. In other words, a rule is either object path specific or interface specific.**

Name	Data type	Required	List of values	Description
ifn	string	no		Interface name. A * indicates a prefix match.
mbr	string	no		Member name
type	string	no	■ M: method call ■ S: signal ■ P: property	Message type. If the type is not specified, the Interface definition will be examined in the following order to determine whether the member name is. 1. A method call or signal. 2. A property.
read-only	boolean	no		Read-only flag applicable to Property only. The default value is false.
mutualAuth	boolean	no		Mutual authorization required. Both peers (local and remote) are required to be granted. Default is yes for signal and no for method call and property.

Signature record

Name	Data type	Required	List of values	Description
issuer	string	yes		Base64 encoding of the issuer's public key
subject	string	yes		Base64 encoding of the subject's public key
algo	integer	yes	TBD	Digital signature algorithm ID.
signature	string	yes		Base64 encoding of the digital signature. The signed data portion includes all data starting from the version fields to the algo field of the signature record.

2.5.2.2 Mapping between message permission and policy permission

The following tables describe the mapping between the message action (send or receive) and the policy permission.

Permission required for Message action

	Provider		Consumer	
Member type		**read-only=false**		**Read-only=false**
property	receive GetProperty	receive SetProperty	send GetProperty	send SetProperty
method call	receive		send	
signal	send		receive	

Enforcing the rules at message creation or receipt

Message action	Required permission		Affected member
	Local peer	Remote peer	
send GetProperty	Consumer	Provider (if mutual authorization is required)	property
receive GetProperty	Provider	Consumer	property
send SetProperty	Consumer read-only=false	Provider read-only=false (if mutual authorization is required)	property
receive SetProperty	Provider	Consumer read-only=false	property
send method call	Consumer	Provider (if mutual authorization is required)	method call
receive method call	Provider	Consumer	method call
send signal	Provider	Consumer	signal
receive signal	Consumer	Provider	signal

2.5.2.3 Search order

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).

The following search order is performed against the authorization rules to find a match. A rule without wild card is stronger than one with wild card. Once a match is found, the search stops.

1. Object path
2. Interface name, member name.
3. Interface name.

2.5.2.4 Materializing the rules

Policies are applied to a peer connection as follows:

1. The ANY-USER policy is applied
2. All guild-in-common policies are applied in undefined order. Per guild-in-common, the materialized authorization rules are the intersection of the authorization rules between the consumer and provider. This intersection will be unioned with the previous result.
3. All peer-specific policies are applied in undefined order. The additional materialized authorization rules will be unioned with the previous result.

2.5.3 Examples

The interface names and members are referenced from the following sample introspection XML.

```
<node
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="https://www.allseenalliance.org/schemas/introspe
ct.xsd">
    <interface name="org.allseenalliance.control.OnOff">
        <method name="On"></method>
        <method name="Off"></method>
    </interface>
    <interface name="org.allseenalliance.control.TV">
        <property name="Channel" type="u" access="readwrite"></property>
        <method name="Up"></method>
        <method name="Down"></method>
        <signal name="ChannelChanged">
           <arg name="newChannel" type="u"/>
        </signal>
    </interface>
    <interface name="org.allseenalliance.control.Mouse">
        <property name="MousePosition" type="(ii)" access="readwrite"/>
        <method name="ClickMouse"></method>
        <method name="WheelMouse">
           <arg name="direction" type="q" direction="in"/>
                <arg name="newMousePosition" type="(ii)" direction="out"/>
        </method>
    </interface>
    <interface name="org.allseenalliance.control.ParentalControl">
        <method name="RateChannel">
           <arg name="channel" type="u" direction="in"/>
           <arg name="rating" type="q" direction="in"/>
        </method>
        <method name="EnableChannel">
           <arg name="channel" type="u" direction="in"/>
        </method>
        <method name="DisableChannel">
           <arg name="channel" type="u" direction="in"/>
        </method>
    </interface>
</node>
```

2.5.3.1 Sample authorization data in a provider policy file

This sample shows a provider authorization data containing an “any-peer” policy and some guild-specific policies.

version	1
provider	peer: type = ANY allow: ifn = org.allseenalliance.control.OnOff peer: type = GUILD, ID = LivingRoomGuildGUID, authority = guildPubKey allow: ifn = org.allseenalliance.control.TV, mbr = Up

	ifn	org.allseenalliance.control.TV	mbr	Down					
	ifn	org.allseenalliance.control.TV	mbr	Channel	type	P	read-only	true	
	ifn	org.allseenalliance.control.Mouse*							

peer	type	GUILD	ID	ParentsGuildGUID	authority	ownerPubKey
allow	obj	/control/settings				

2.5.3.2 Sample authorization data in a membership certificate for a provider

This sample shows a provider can emit a signal to members of the guild.

version	6								
provider	allow	ifn	org.allseenalliance.control.TV	mbr	ChannelChanged	type	S		

2.5.3.3 Sample authorization data for a consumer membership certificate

This sample shows the holder of the certificate has all the granted permissions provided in the provider policy. The guild ID comes from the membership certificate.

version	32		
consumer	allow	ifn	*

2.5.3.4 Sample authorization data for a consumer that restricts some access

This sample shows the holder of the membership certificate has all the granted permissions provided in the provider policy for the guild except a particular interface.

version	9		
consumer	allowAllExcept	ifn	org.allseenalliance.control.Mouse*

2.5.3.5 Sample authorization data in a consumer policy file

This sample shows a consumer authorization data. In this policy, the consumer is not allowed to receive any signal with the exception of receiving a channel changed signal from any member of the guild LivingRoomGuild.

version	8743												
consumer	peer	type	ANY										
	allowAllExcept	type	S										
	peer	type	GUILD	ID	LivingRoomGuildGUID			authority	guildPubKey				
	allow	ifn	org.allseenalliance.control.TV			mbr	ChannelChanged	type	S				

2.5.3.6 Sample authorization data for a specific peer

This sample shows the holder of the access certificate can interact with the specific peer with the specific permissions.

version	77372				
consumer	peer	type	DSA	ID	D24174252B1DD243E0AAF903DE886ECA3FF38C973EAD6B81344F809200A3723900000000BB153866208721F15DC29FA2945266937096B2BAC1FB90EBDA943B2FFDAD2BDE0000000000000000
	allow	ifn	Org.allseenalliance.control.ParentControl		

2.5.3.7 Sample authorization data for creating an admin

This sample shows how to create an admin.

version	435				
admin	peer	type	DSA	ID	D24174252B1DD243E0AAF903DE886ECA3FF38C973EAD6B81344F809200A3723900000000BB153866208721F15DC29FA2945266937096B2BAC1FB90EBDA943B2FFDAD2BDE0000000000000000

2.5.4 Policy Templates (Work-In-Progress)

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

- Template name
- Description
- List of permission rules

2.5.4.1 Example of a Policy Template

This example shows an example of a policy template provided by a TV application:

name	Regular								
desc	Regular access								
provider	allow	ifn	org.allseenalliance.control.OnOff						
		ifn	org.allseenalliance.control.TV	mbr	Up				
		ifn	org.allseenalliance.control.TV	mbr	Down				
		ifn	org.allseenalliance.control.TV	mbr	Channel	type	P	read-only	true
		ifn	org.allseenalliance.control.Mouse*						

2.6 Certificates (Work-in-progress)

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. The subsections will be updated with the X.509 required fields.

2.6.1 Policy certificate

Table 2-2 lists the policy certificate fields. This type of certificate and its authorization data can be used as a policy certificate or a peer specific access certificate.

Table 2-2. Policy certificate fields

Field name	Description
version	■ version is 1. ■ ECC curve is NIST P-256 ■ External Data digest algorithm is SHA-256. ■ DSA algorithm is ECC NIST P-256 DSA.

Field name	Description
issuer	Issuer public key.
subject	Subject field holding the certificate holder's public key.
validityFrom	Validity period. Subfield Valid From. It's represented in seconds since EPOCH Jan 1, 1970.
validityTo	Validity period. Subfield ValidTo. It's represented in seconds since EPOCH Jan 1, 1970.
delegate	Delegate flag. Must be set to false.
digest	Digest of the authorization data.
sig	DSA signature, which is computed over the fields from subject field to digest field by the issuer.

2.6.2 Membership certificate

Table 2-3 lists the guild-specific certificate fields. This type of certificate and its authorization data can be used a membership certificate.

Table 2-3. Guild-specific certificate fields

Field name	Description
version	■ version is 2. ■ ECC curve is NIST P-256 ■ External Data digest algorithm is SHA-256. ■ DSA algorithm is ECC NIST P-256 DSA.
issuer	Issuer public key.
subject	Subject field holding the certificate holder's public key.
validityFrom	Validity period. Subfield Valid From. It's represented in seconds since EPOCH Jan 1, 1970.
validityTo	Validity period. Subfield ValidTo. It's represented in seconds since EPOCH Jan 1, 1970.
delegate	Delegate flag.
guild	Guild ID
digest	Digest of the authorization data.
sig	DSA signature, which is computed over the fields from subject field to digest field by the issuer.

2.6.3 User equivalence certificate

Table 2-4 lists the user equivalence certificate fields. The subject will have all the privileges as the issuer.

Table 2-4. User equivalence certificate fields

Field name	Description
version	■ version is 3. ■ ECC curve is NIST P-256 ■ External Data digest algorithm is SHA-256. ■ DSA algorithm is ECC NIST P-256 DSA.
issuer	Issuer public key.
subject	Subject field holding the certificate holder's public key.
validityFrom	Validity period. Subfield Valid From. It's represented in seconds since EPOCH Jan 1, 1970.
validityTo	Validity period. Subfield ValidTo. It's represented in seconds since EPOCH Jan 1, 1970.
delegate	Delegate flag. Must be set to false.
sig	DSA signature, which is computed over the fields from subject field to delegate field by the issuer.

2.6.4 Identity certificate

Table 2-5 lists the Identity certificate fields.

Table 2-5. Identity certificate fields

Field name	Description
version	■ version is 4. ■ ECC curve is NIST P-256 ■ External Data digest algorithm is SHA-256. ■ DSA algorithm is ECC NIST P-256 DSA.
issuer	Issuer public key.
subject	Subject field holding the certificate holder's public key.
validityFrom	Validity period. Subfield Valid From. It's represented in seconds since EPOCH Jan 1, 1970.
validityTo	Validity period. Subfield ValidTo. It's represented in seconds since EPOCH Jan 1, 1970.
delegate	Delegate flag. Must be set to false.
aliasLen	Length of the alias. The maximum length allowed is 40 bytes.
alias	Byte array for the alias.
digest	Digest of the identity data.
sig	DSA signature, which is computed over the fields from subject field to digest field by the issuer.

2.6.4.1 Identity data format

The suggested Identity information can be expressed as vCard data using JSON format as described in RFC 7095.

2.6.5 Guild equivalence certificate

Table 2-6 lists the guild equivalence certificate fields.

Table 2-6. Guild equivalence certificate fields

Field name	Description
version	■ version is 5. ■ ECC curve is NIST P-256 ■ External Data digest algorithm is SHA-256. ■ DSA algorithm is ECC NIST P-256 DSA.
issuer	Issuer public key.
subject	Subject field holding the certificate holder's public key.
validityFrom	Validity period. Subfield Valid From. It's represented in seconds since EPOCH Jan 1, 1970.
validityTo	Validity period. Subfield ValidTo. It's represented in seconds since EPOCH Jan 1, 1970.
delegate	Delegate flag. Must be set to false.
guild	Locally definedguild ID. All membership certificates issued by the subject or its delegates will be treated as equivalences the locally definedguild.
sig	DSA signature, which is computed over the fields from subject field to guild field by the issuer.

2.7 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.

2.7.1 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

2.7.2 Users set up by Dad

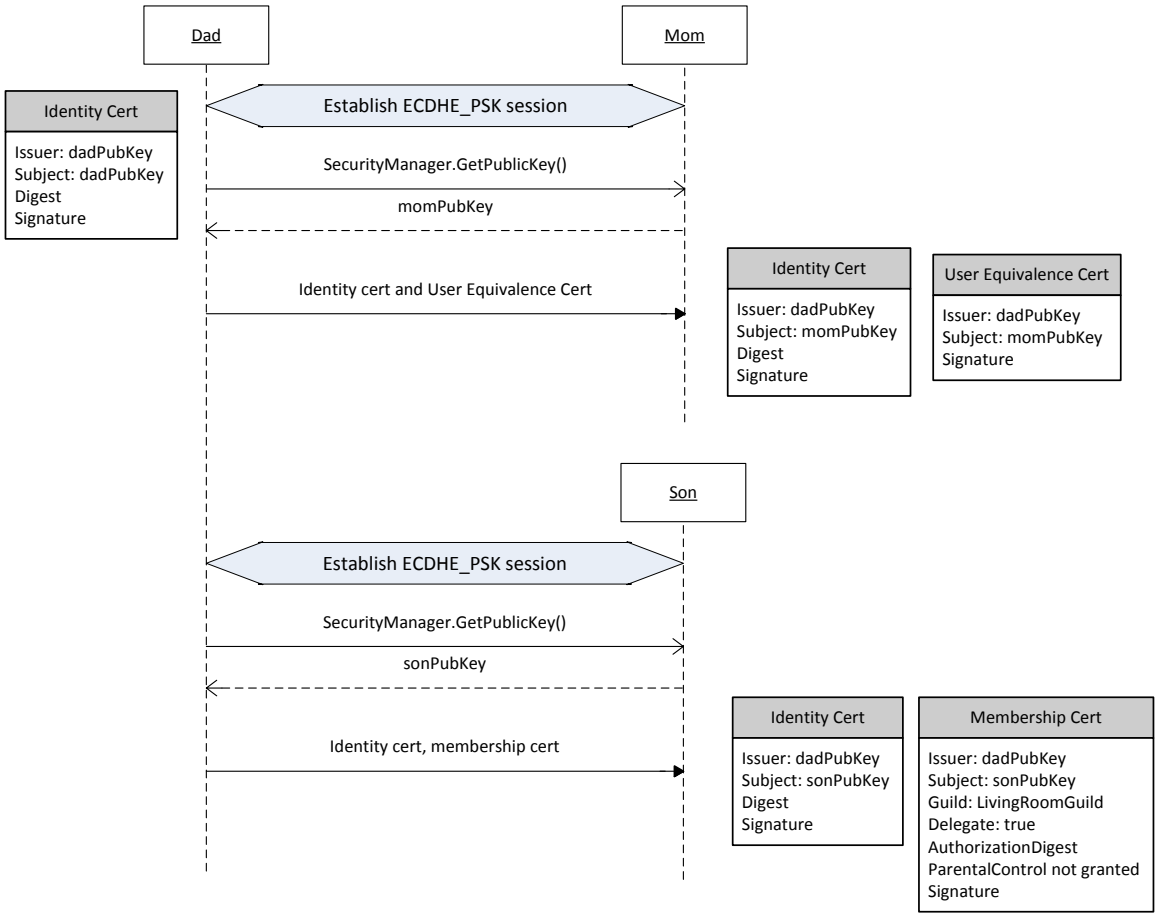

Figure 2-16. Use case - users set up by Dad

2.7.3 Living room set up by Dad

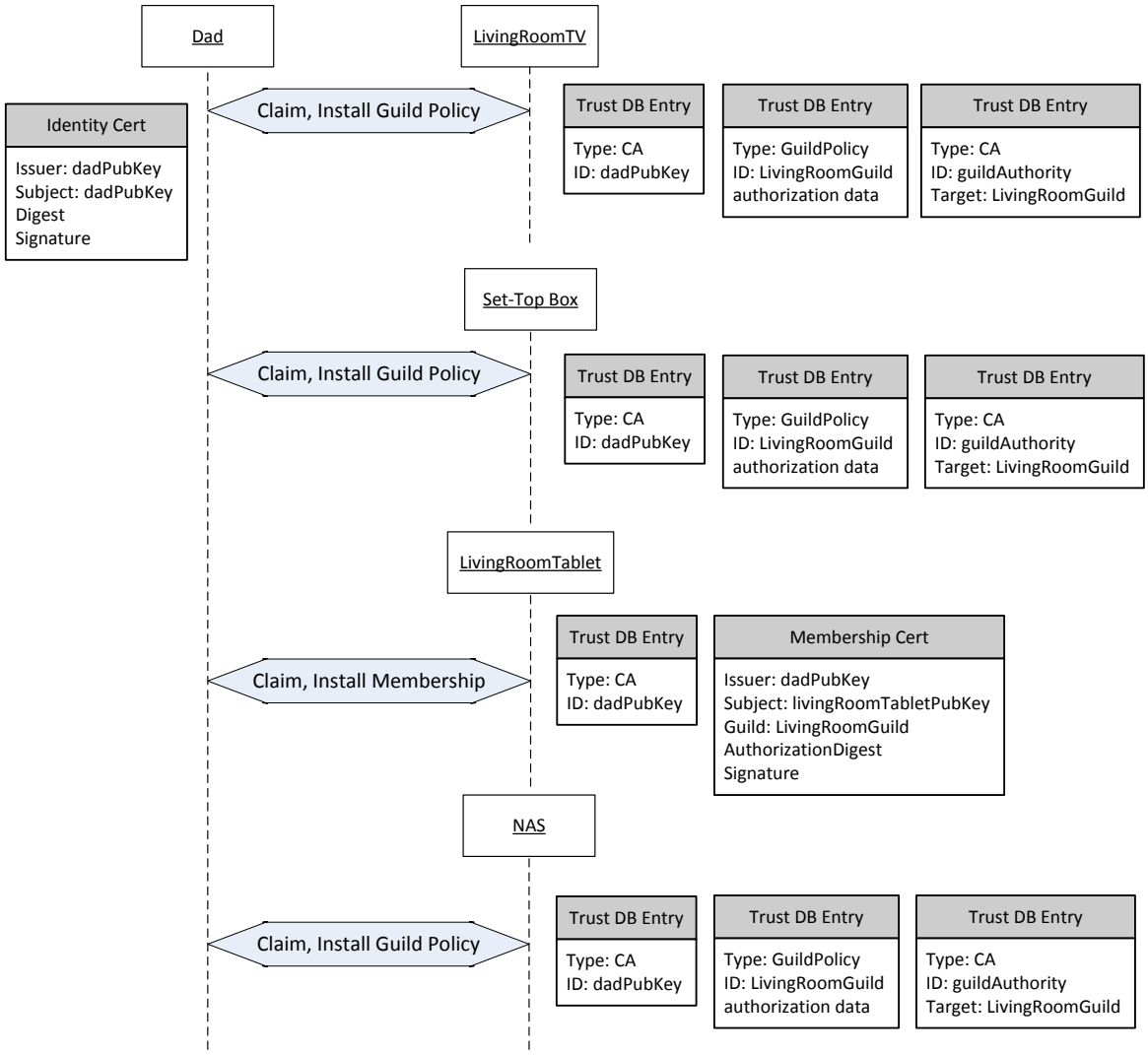

Figure 2-17. Use case - living room set up by Dad

2.7.4 Son's bedroom set up by son

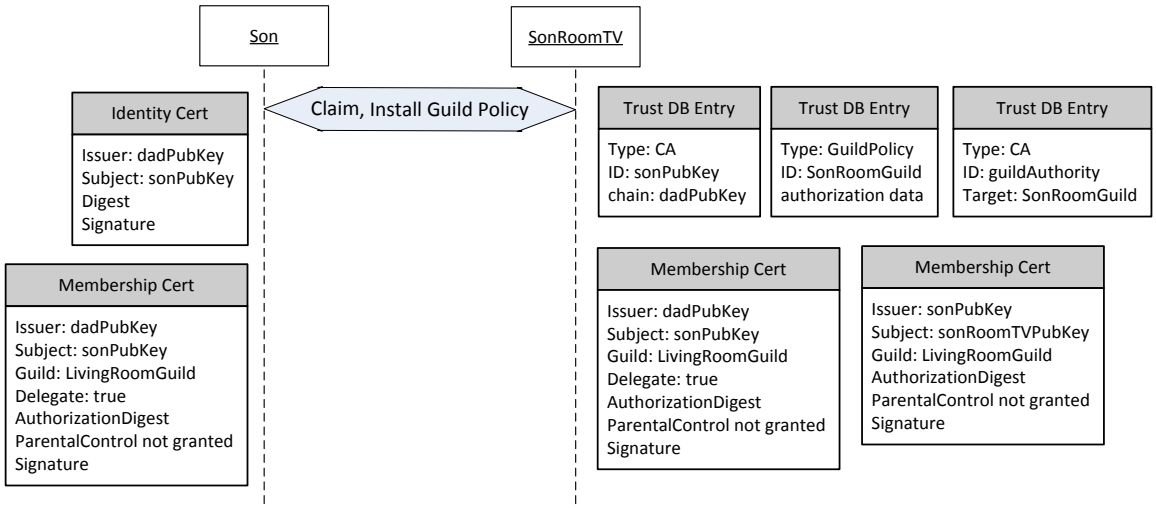

Figure 2-18. Use case - son's bedroom set up by son

2.7.5 Master bedroom set up by Dad

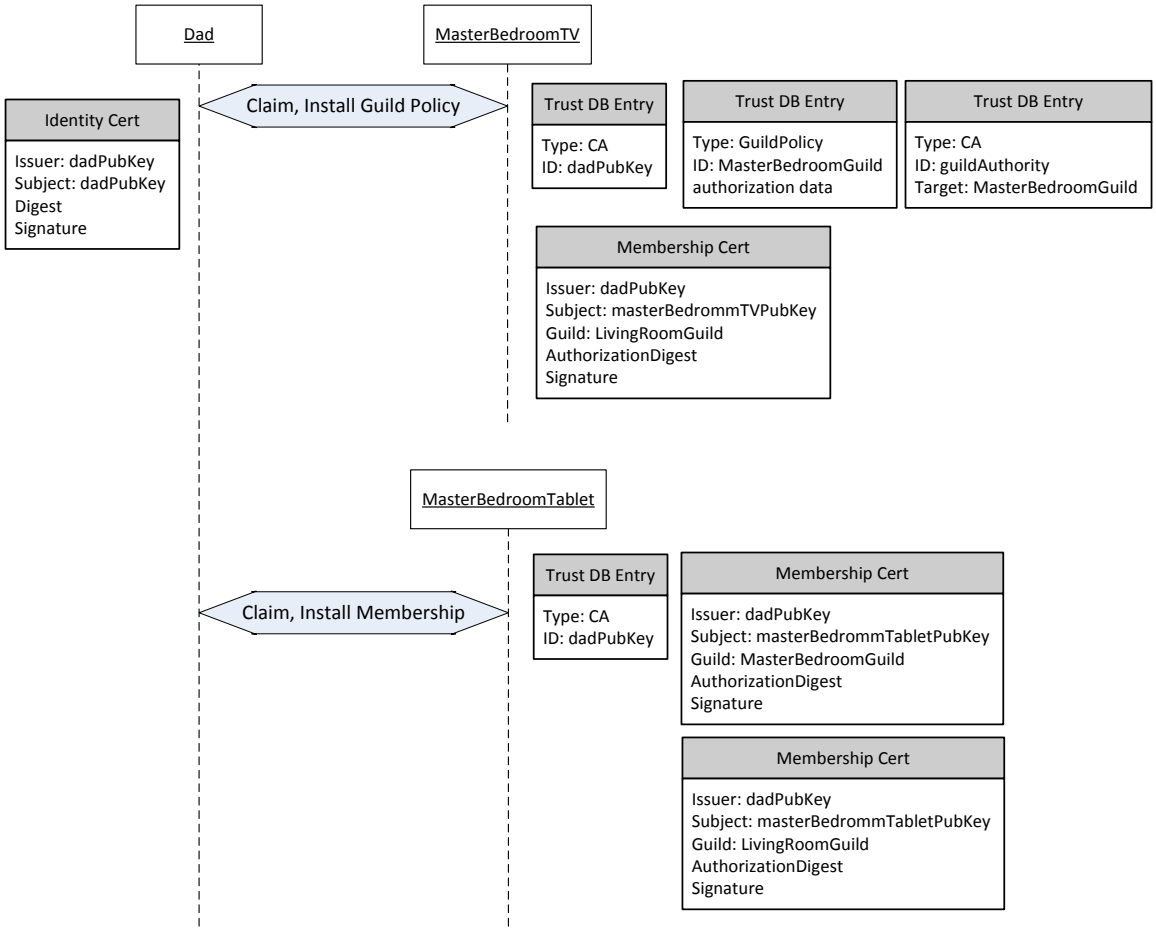

Figure 2-19. Use case - master bedroom set up by Dad

2.7.6 Son can control different TVs in the house

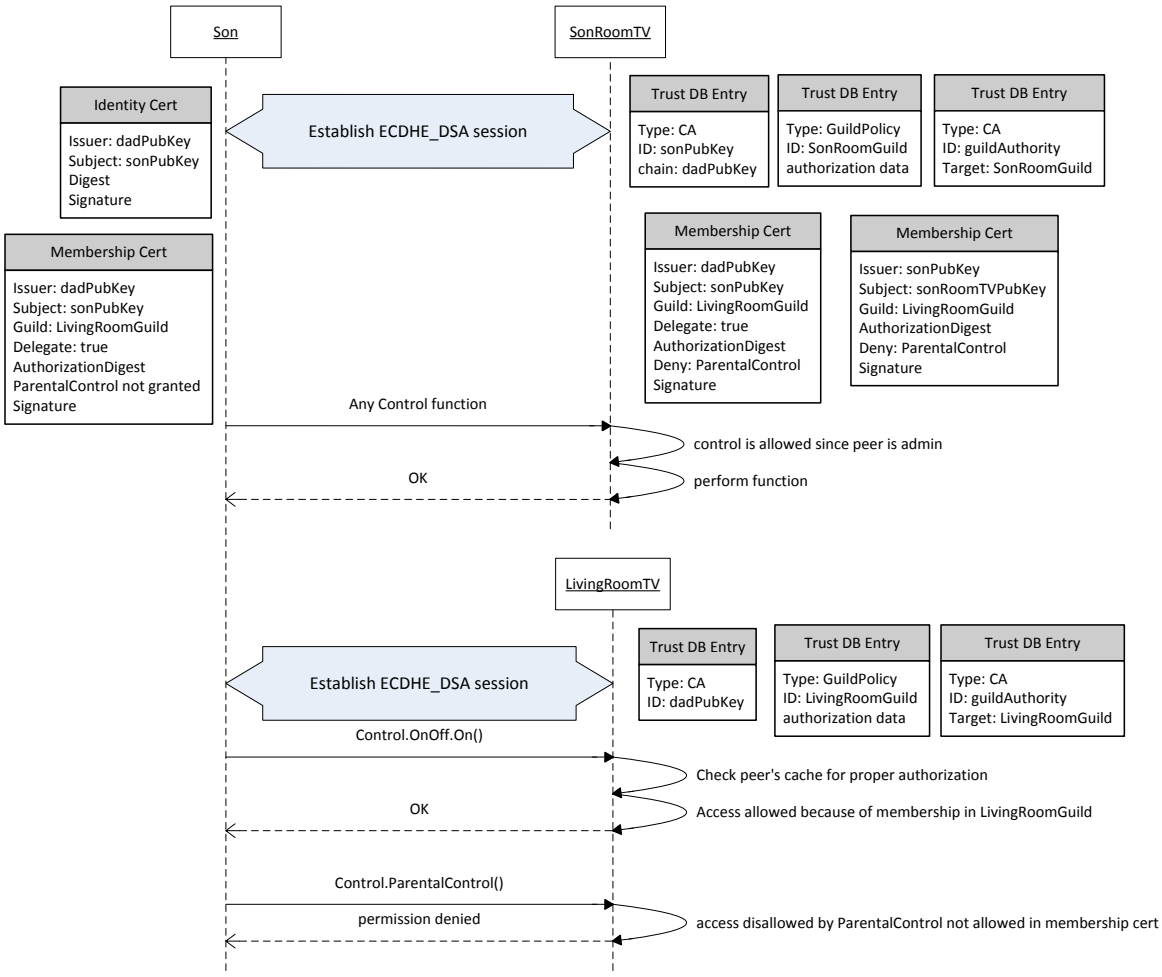

Figure 2-20. Use case – Son can control different TVs in the house

2.7.7 Living room tablet controls TVs in the house

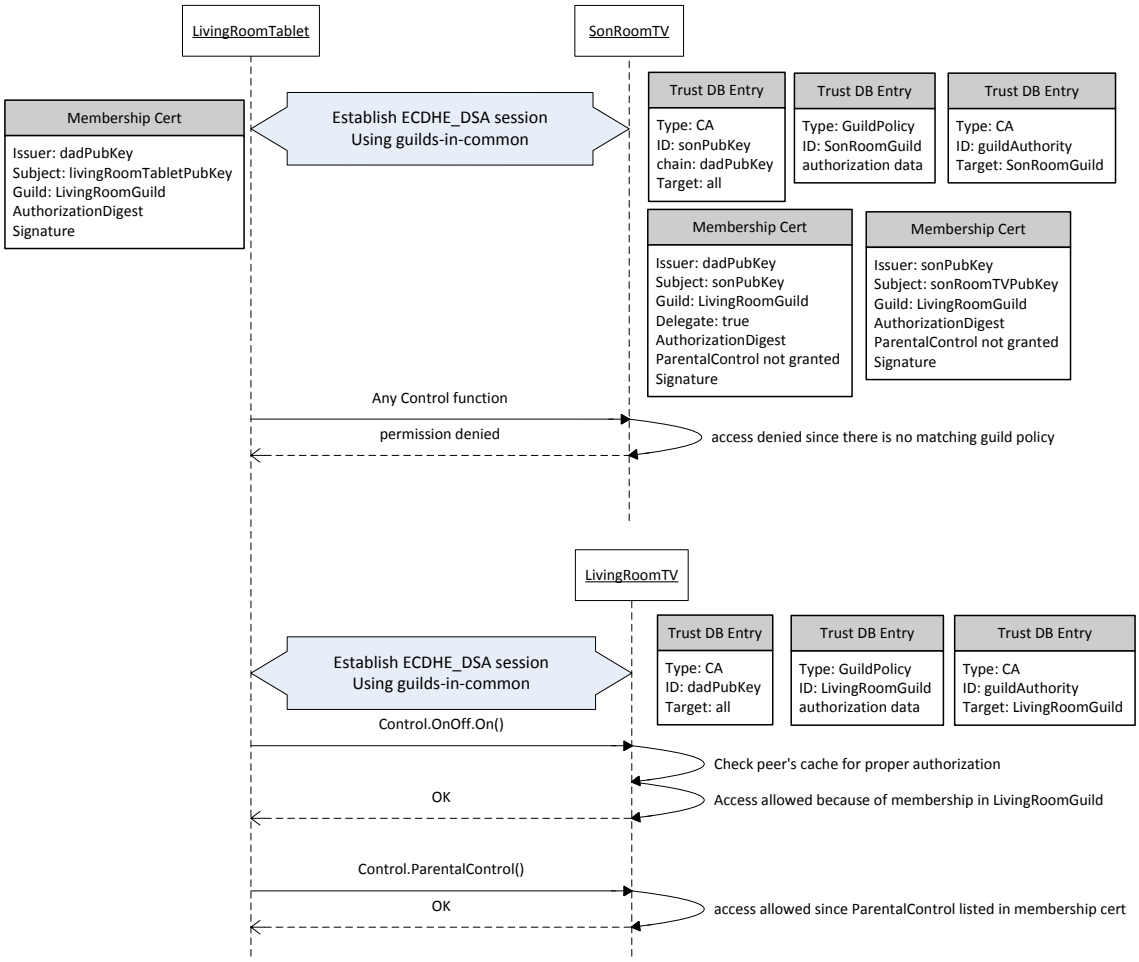

Figure 2-21. Use case - Living room tablet controls TVs

3 Enhancements to Existing Framework

3.1 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 , RFC6655, and 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 • SPKI based certificate	• Standard Client • Thin Client • Deprecated
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 TSL 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
TLS_RSA_WITH_AES_128_CCM_8	• SHA256 • X.509 certificate	• Standard Client	6655

3.1.1 Add a Claimable Field to the About Announcement

A new number field named **claimable** will be added to the About Announcement to show the claim state of the application. The possible values of this field are:

- 0 -- not claimable
- 1 – claimable
- 2 -- claimed

4 Future Considerations

4.1 Broadcast signals and multipoint sessions

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